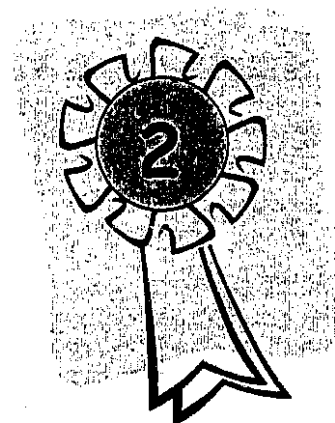

Dana Lica

Mircea Pașoi

INFORMATICĂ

FUNDAMENTELE PROGRAMĂRII

Culegere de probleme – Pascal și C/C++
pentru clasa a X-a



Editura L&S Info-mat

Toate drepturile asupra acestei lucrări aparțin editurii L&S INFO-MAT.

Reproducerea integrală sau parțială a textului din această carte este posibilă doar cu acordul în scris al editurii L&S INFO-MAT.

Editura L&S INFO-MAT:

Adresa: Str. Stânjeneilor nr. 8, bl. 29, sc. A, et. 1, apt. 12, sector 4, București.

Telefon: 021-3321315; 021-6366344; 0722-530390; 0722-573701;

Fax: 021-3321315;

E-mail: tsorin@ls-infomat.ro;

Web Site: www.ls-infomat.ro.

**Descrierea CIP a Bibliotecii Naționale a României
LICA, DANA**

Informatică : fundamentele programării / Dana Lica,
Mircea Pașoi. – București, Editura L&S Soft, 2005-
3 vol.

ISBN 973-86022-9-7

Vol. 2 : Culegere de probleme – Pascal și C++
pentru clasa a X-a. – București : Editura L&S Infomat,
2005. – ISBN 973-7658-02-7

I.Pașoi, Mircea

004.42 PASCAL(075.35)(076)

004.42 3C(075.35)(076)

004.42 C++(075.35)(076)

Tiparul executat la S.C. LUMINA TIPO s.r.l.

Str. Luigi Galvani nr. 20 bis, sect. 2, București, tel./fax 211.32.60

Tel. 212.29.27, e-mail: office@luminatipo.com, www.luminatipo.com.

Cuprins

Capitolul 1

Tipuri structurate de date

1.1 Șir de caractere.....	5
1.1.1 Teste cu alegere multiplă și duală.....	5
1.1.2 Probleme rezolvate.....	14
1.1.3 Probleme propuse.....	25
1.2 Înregistrare - structură.....	33
1.2.1 Teste cu alegere multiplă și duală.....	33
1.2.2 Probleme rezolvate.....	39
1.2.3 Probleme propuse.....	48
1.3 Probleme de concurs ce procesează date structurate.....	51
1.3.1 Probleme rezolvate.....	51
1.3.2 Probleme propuse.....	73

Capitolul 2

Subprograme definite de utilizator

2.1 Subprograme implementate în manieră iterativă.....	91
2.1.1 Teste cu alegere multiplă și duală.....	91
2.1.2 Probleme rezolvate.....	101
2.1.3 Probleme propuse.....	118
2.2 Subprograme implementate în manieră recursivă.....	129
2.2.1 Teste cu alegere multiplă și duală.....	129
2.2.2 Probleme rezolvate.....	140
2.2.3 Probleme propuse.....	158
2.3 Probleme de concurs.....	166
2.3.1 Probleme rezolvate.....	166
2.3.2 Probleme propuse.....	189

<u>Indicații și răspunsuri</u>	203
--------------------------------------	-----

Tipuri de Date Structurate

1.1. Șir de caractere

1.1.1 Teste cu alegere multiplă și duală

1. Care dintre următoarele instrucțiuni sunt corecte sintactic? Variabilele care intervin sunt de tip *string* / *char* *.

- | | |
|----------------------------|---------------------------|
| a) pos('a',s):=2; | a) p=strchr('a',b); |
| b) c:=delete(s,2,2); | b) k = strchr(s,x)-s; |
| c) writeln(length('ana')); | c) cout<<strlen("ana"); |
| d) insert('red',s,4); | d) k = strcat("ma","ma"); |

2. Se consideră variabila *s* de tip *string* / *char* *. Care dintre următoarele secvențe afișează valoarea variabilei *s* din care lipsesc primul și ultimul caracter? Variabila *i* aparține unui tip întreg.

- | | |
|---|---|
| a) delete(s,1,1);
delete(s,length(s),1);
writeln(s); | a) strcpy(s,s+1);
strcpy(s+strlen(s)-1,
s+strlen(s));
cout<<s<<endl; |
| b) for i:=2 to length(s)-1 do
write(s[i]); | b) for(i=1; i<strlen(s)-1; i++)
cout<<s[i]; |
| c) n:=length(s);
delete(s,1,1);
delete(s,n,1);
writeln(s); | c)
strcpy(s,strchr(s,s[strlen(s)-1])+1);
strcpy(s,s+1);
cout<<s; |
| d) write(copy(s,2,length(s))); | d) for(i=0; i<strlen(s)-2; i++)
cout<<s[i]; |

3. Ce se va afișa în urma executării următoarei secvențe de instrucțiuni?

- | | |
|--|--|
| s:='Primavara';
for i:=1 to 3 do
delete(s,2,1);
writeln(s); | a = "Primavara";
for (int i=1;i<=3;i++)
strcpy(a+1,a+2);
cout<<a; |
|--|--|

- | | | | |
|------------|---------|---------|----------|
| a) Pavara; | b) ara; | c) rim; | d) Para. |
|------------|---------|---------|----------|

12. Considerăm următoarele declarații:

```
var x,y,z:string[200];
```

```
char* z;  
char x[200],y[200];
```

Identificați expresiile corecte sintactic din lista următoare:

- | | |
|----------------------------------|----------------------------------|
| a) x := '12' + 'zile'; | a) z = z++; |
| b) z := y + x; | b) strcat(x,y); |
| c) y := y + str(13, x); | c) z = {x!=y && y!=z}; |
| d) x := dec(z); | d) x = "12Azile"; |
| e) x := copy('12Azile',4,2); | e) z = strchr("exercitii", ' '); |
| f) x := insert('dimineata',y,3); | f) z = strcpy('A',"BBB"); |

13. Considerăm următoarea secvență de program, în care x și y sunt variabile din tipul șir de caractere. Ce se va afișa în urma execuției lor?

```
x:='dimineata';  
y:='min' + x[length(x)];  
writeln(pos(y,x));
```

```
x="dimineata"; y="min";  
strcat(y,x+strlen(x)-1);  
p=strstr(y,x);  
cout<<(p!=NULL) ? (p-y) : 0;
```

- | | | | |
|------|------|-------|----------|
| a) 0 | b) 3 | c) ta | d) dieta |
|------|------|-------|----------|

14. Considerăm următoarele declarații:

```
var a:string[200]; i:byte;
```

```
char a[200]; unsigned int i;
```

Se știe că șirul de caractere a conține numai caractere distincte, excepție făcând ultimele două care sunt identice. Identificați care dintre secvențele următoare de instrucțiuni este echivalentă cu funcția length() / strlen()?

- | | |
|---|---|
| a) i:=1;
while a[i]<>a[i+1] do inc(i);
writeln(i+1); | a) i:=0;
while(a[i] != a[i+1]) i++;
cout<<i+2; |
| b) i:=0;
while a[i]<>a[i+1] do inc(i);
writeln(i); | b) i:=0;
while(a[i] != a[i+1]) i++;
cout<<i+1; |
| c) i:=1;
if a[i] = a[i+1] then write(i) | c) i:=0;
if (a[i] == a[i+1]) cout<<i; |
| d) i:=1;
if a[i]<>a[i+1] then inc(i)
else write(i+1); | d) i:=0;
if (a[i] != a[i+1]) i++ ;
else cout<<i ; |

15. Considerăm următoarele declarații:

```
var a,b: string[100];
```

```
char *a, *b;
```

Ce se va afișa în urma execuției următoarei secvențe de instrucțiuni?

```
a := 'mama'; b := 'Mamaie';  
if a>b then write(a)  
else  
if a=b then write('identice')  
else write(b);
```

```
a = "mama"; b="Mamaie";  
if (strcmp(a,b)>0) cout<<a;  
else  
if (strcmp(a,b)==0)  
cout<<"identice";  
else cout<<b;
```

- a) Mamaie;
b) mama;
c) identice;
d) Secvența de instrucțiuni propusă nu execută nici o afișare.

16. Considerăm următoarele declarații:

```
var a:array[1..9] of string[20];  
n,i,j:byte; x:string[20];
```

```
char a[20][20]; unsigned int n,i,j;  
char x[20];
```

și următoarea secvență de program:

```
readln(n);  
for i:=1 to n do readln(a[i]);  
for i:=1 to n-1 do  
for j:=i+1 to n do  
if a[j]<a[i] then begin  
x:=a[j]; a[j]:=a[i]; a[i]:=x;  
end;
```

```
cin>>n;  
for(i:=0;i<n;i++) cin>>a[i];  
for (i:=0;i<n-1;i++)  
for (j:=i+1;j<n;j++)  
if (strcmp(a[j],a[i])<0) {  
strcpy(x,a[j]); strcpy(a[j],a[i]);  
strcpy(a[i],x); }
```

Ce prelucrare realizează această secvență asupra elementelor vectorului a?

- a) Ordonează crescător elementele tabloului a după lungimea șirurilor de caractere;
b) Ordonează lexicografic crescător elementele tabloului a;
c) Înlocuiește elementele tabloului cu șirul de caractere maxim din punct de vedere lexicografic;
d) Înlocuiește elementele tabloului cu șirul de caractere de lungime maximă.

17. Considerăm programul următor:

```
var a,b: string;  
i,x,y:integer;  
begin  
readln(a);readln(b);  
i:=1;  
x:=length(a); y:=length(b);  
while (y-x+1>=i)  
and(copy(b,i,x)<>a) do inc(i);  
if i > y-x+1 then i:=0;  
writeln(i);  
end.
```

```
#include <iostream.h>  
#include <string.h>  
char a[256],b[256]; int i,x,y;  
void main() {  
cin>>a; cin>>b;  
x=strlen(a); y=strlen(b);  
for(i:=0; y-x>=i &&  
strncmp(b+i,a,x)!=0; i++)  
if (i>y-x) i=-1;  
cout<<i<<endl;  
}
```

Identificați care dintre expresiile următoare sunt echivalente cu de programul de mai sus (la evaluarea expresiei se obține aceeași valoare cu cea afișată prin program).

- | | |
|-------------------|-------------------|
| a) concat(a,b); | a) strlen(a); |
| b) length(a + b); | b) strlen(b); |
| c) pos(a,b); | c) strstr(b,a)-b; |
| d) copy(a,b,i); | d) strcmp(a,b); |

18. Fie secvența de instrucțiuni următoare :

<pre> For i :=1 to length(a) do if a[i] in ['A'..'Z'] then a[i] :=chr(ord(a[i])+32) ; </pre>	<pre> for(i=0;i<strlen(a);i++) if (a[i]>='A' && a[i]<='Z') a[i] += 32; </pre>
--	--

Știind că a este un șir de caractere și i o variabilă de tip întreg, identificați prelucrarea realizată asupra caracterelor sale.

- a) transformarea caracterelor de tip minusculă în majuscula corespunzătoare;
- b) inserarea șirului de caractere 32 după fiecare caracter de tip majusculă.
- c) transformarea caracterelor de tip majusculă în minuscula corespunzătoare;
- d) ordonarea alfabetică a majusculilor în cadrul șirului;
- e) ștergerea caracterelor spațiu.

19. Fie următorul program:

<pre> var s:string[10]; i:integer; x:char; begin s:='clasa'; for i:=1 to length(s)-1 do begin if(s[i]>s[i+1]) then begin x:=s[i]; s[i]:=s[i+1]; s[i+1]:=x; end; write(s, ' '); end; end. </pre>	<pre> #include<iostream.h> #include<string.h> char s[10];int i; char x; void main() { strcpy(s,"clasa"); for(i=0;i<strlen(s)-1;i++){ if(s[i]>s[i+1]){ x=s[i]; s[i]=s[i+1]; s[i+1]=x; } cout<<s<<" "; } } </pre>
--	---

Ce se va afișa pe ecran în urma execuției acestui program?

- a) clasa clasa calas calas
- b) clasa calsa calas
- c) clasa calsa calsa calas
- d) clasa aclsa aclas acals

20. Se consideră șirurile de caractere a și b . Identificați care dintre următoarele secvențe de instrucțiuni modifică valoarea șirului b prin ștergerea primei apariții a lui a în b . În situația în care a nu se regăsește în b , valoarea acestuia din urmă trebuie să rămână neschimbată. Variabila x este de tip *integer* pentru Pascal, respectiv de tip *int* pentru C/C++.

- | | |
|---|---|
| <pre> a) delete(b,pos(a,b),length(a)); b) x:=pos(a,b); b:=copy(b,1,x-1)+ copy(b,x+length(a),x) c) delete(a,pos(a,b),length(b)); d) x:=pos(a,b); if x<>0 then b:=copy(b,1,x-1)+ copy(b,x+length(a),x); </pre> | <pre> a) x=strstr(b,a)-b; if (x>=0) strcpy(b+x,b+x+strlen(a)); b) x=strstr(b,a)-b; strcpy(b+x,b+x+strlen(a)-1); c) x=strstr(a,b)-a; if (x>=0) strcpy(a+x,a+x+strlen(b)); d) x=strstr(a,b)-a; strcpy(a+x,a+x+strlen(b)); </pre> |
|---|---|

21. Considerăm următorul program. Ce condiție trebuie îndeplinită pentru ca în urma rulării acestuia să se afișeze mesajul *Corect* ?

<pre> var a,b:string; begin readln(a); readln(b); while(a[1]=b[1])and(a<>'')and (b<>'')do begin delete(a,1,1); delete(b,1,1); end; if(a='')and(b='') then write('Corect') else write('Incorect'); end. </pre>	<pre> #include<iostream.h> #include<string.h> char a[256],b[256]; void main() { cin>>a; cin>>b; while(a[0]==b[0]&&a[0]!=0 &&b[0]!=0){ strcpy(a,a+1); strcpy(b,b+1); } if (a[0]==0 && b[0]==0) cout<<"Corect"; else cout<<"Incorect"; } </pre>
---	---

- a) afișează mesajul *Corect* doar dacă șirurile a și b au lungimi egale;
- b) afișează mesajul *Corect* doar dacă șirurile a și b au valori identice;
- c) afișează mesajul *Corect* doar dacă șirurile a și b au lungimi nule;
- d) afișează mesajul *Corect* doar dacă șirurile a și b au valori identice de lungime 1;

22. Se consideră următorul program:

```
var a,b:string;
    i:integer;
begin
    b:='';
    for i:=1 to 3 do begin
        readln(a);
        b:=b+copy(a,i,length(a)-i+1);
    end;
    writeln(b)
end.
```

```
#include<iostream.h>
#include<string.h>
char a[256],b[256]; int i;
void main()
{ strcpy(b,"");
  for(i=0; i<3; i++){
    cin>>a; strcat(b,a+i);
  }
  cout<<b;
}
```

Identificați ce se va afișa dacă de la tastatură se vor introduce, în ordine, șirurile de caractere: "copil", "masina", "bloc".

- a) opilslinac b) cmb c) copilmasinbl d) copilasinaoc

23. Considerăm următorul program:

```
var a,b:string;
begin
    readln(a);
    readln(b);
    while (pos(a[1],b)<>0) do
    begin
        delete(b,pos(a[1],b),1);
        delete(a,1,1);
    end;
    if (a='') and (b='') then
        write('Da')
    else
        write('nu');
end.
```

```
#include<iostream.h>
#include<string.h>
char a[256],b[256],*p; int i;
void main()
{ cin>>a; cin>>b;
  while (strchr(b,a[0])!=NULL &&
        a[0]!='\0') {
    p=strchr(b,a[0]);
    strcpy(p,p+1);strcpy(a,a+1);
  }
  if (a[0]=='\0' && b[0]=='\0')
    cout<<"da";
  else cout<<"nu";
}
```

La sfârșitul rulării acestuia se va afișa mesajul *Da* dacă și numai dacă:

- a) șirurile *a* și *b* au valori egale;
 b) șirurile *a* și *b* au lungimi identice;
 c) șirurile *a* și *b* sunt formate din exact aceleași caractere, eventual în altă ordine.
 d) fiecare caracter al șirului *a* apare și în șirul *b*;

24. Fie *a* un șir de caractere (*string* / *char* *) și *x* o variabilă întreagă. Care dintre următoarele instrucțiuni elimină toată secvența de caractere identice de la începutul șirului *a* (pentru a fi ștersă, secvența va conține minimum 2 caractere):

```
a) x:=length(a);
   while ((a[1])=a[2]) and
     (length(a)>1) do delete(a,1,1);
   if length(a)<>x then
     delete(a,1,1);
```

```
a) x=strlen(a);
   while (*a==*(a+1)) {a=a++;}
   if (strlen(a)!=x)
     strcpy(a,a+1);
```

```
b) while ((a[1])=a[2]) and
  (length(a)>1) do delete(a,1,1);
c) while ((a[1]) <> a[2]) and
  (length(a)>1) do delete(a,1,1);
d) while ((a[1]) <> a[2]) and
  (length(a)>1) do delete(a,1,1);
   delete(a,1,1);
```

```
b) while (*a==*(a+1)) {
    a=a++;
  }
  strcpy(a,a+1);
c) while (*a!=*(a+1)) {a=a++;}
  strcpy(a,a+1);
d) while (*a==*(a+1)) {a=a++;}
  strcpy(a,a+1);
```

25. Fie *a* un șir de caractere (*string* / *char* *) și *i*, *j* variabile întregi. Care dintre următoarele instrucțiuni permit afișarea mesajului "Da" dacă și numai dacă valoarea lui *a* este palindrom?

```
a) i:=1; j:=length(a);
   while (a[i]<>a[j]) and (i<j) do
   begin dec(i); inc(j); end;
   if (i>j) then write('DA')
   else write('NU');
```

```
a) i:=0; j:=strlen(a)-1;
   while ((a[i]==a[j]) && (i<j))
   { i--; j++; }
   if (i>=j) cout<<"DA";
   else cout<<"NU";
```

```
b) i:=1; j:=length(a);
   while (a[i]<>a[j]) and (i<j) do
   begin inc(i); dec(j); end;
   if (i>=j) then write('DA')
   else write('NU');
```

```
b) i:=0; j:=strlen(a)-1;
   while ((a[i]!=a[j]) && (i<j))
   { i++; j--; }
   if (i<=j) cout<<"DA";
   else cout<<"NU";
```

```
c) i:=1; j:=length(a);
   while (a[i]=a[j]) and (i<j) do
   begin inc(i); dec(j); end;
   if (i>=j) then write('DA')
   else write('NU');
```

```
c) i:=0; j:=strlen(a)-1;
   while ((a[i]==a[j]) && (i<j))
   { i++; j--; }
   if (i>=j) cout<<"DA";
   else cout<<"NU";
```

```
d) i:=1; j:=length(a);
   while (a[i]=a[j]) and (i<j) do
   begin dec(i); inc(j); end;
   if (i>=j) then write('DA')
   else write('NU');
```

```
d) i:=0; j:=strlen(a)-1;
   while ((a[i]!=a[j]) && (i<j))
   { i--; j++; }
   if (i>=j) cout<<"DA";
   else cout<<"NU";
```

26. Știind că variabila *a* este folosită pentru a memora, ca șir de caractere, numele unei discipline studiate în liceu (maximum 50 caractere), identificați o declarație corectă a sa:

```
a) var a:string;
b) var a:string[39];
c) var a:string[50];
d) var a:array[1..50] of string;
```

```
a) char a;
b) char a[39];
c) char a[50];
d) char* a[20];
```

27. Știind că variabila *a* este utilizată pentru a memora numele celor 7 zile ale săptămânii, cum trebuie ea declarată:

a) var a: string[7][7];	a) char a[7];
b) var a: array[1..7] of string;	b) char a[7][17];
c) var a: string[7];	c) char a;
d) var a: array[1..7] of char;	d) char** a[7];

28. Considerăm secvența de instrucțiuni următoare în care variabila *s* este un șir de caractere, *i* și *k* variabile întregi, *x* o variabilă din tipul *char* iar *ok* este o variabilă din tipul *boolean*(Pascal) - *int*(pentru C/C++):

<pre> i := 1; ok := true; k := 0; while (i<=length(s))and ok do begin if s[i]=x then begin k:=i; ok:=false; end; inc(i); end; </pre>	<pre> i = 0; ok = 1; k = 0; while(i<strlen(s) && ok){ if(s[i]==x) { k = i; ok = 0; }i++;} </pre>
---	---

Care dintre atribuirile următoare conduc la obținerea aceleiași valori pentru *k*, ca cea obținută în urma execuției secvenței prezentate ?

a) k := concat(s,x)	a) k = strcat(s,x)
b) k := length(s)	b) k = strlen(s)
c) k := pos(x,s)	c) k = strchr(s,x)-s
d) k := pos(s,x)	d) k = strchr(s,x)-x

1.1.2 Probleme rezolvate

1. Se consideră un text în care unicul separator este spațiul. Știind că între oricare două cuvinte pot exista mai mulți separatori, să se determine numărul de cuvinte din text.

Exemplu: Pentru textul 'Am venit repede' se va afișa 3.

Soluție

Algoritmul presupune parcurgerea caracter cu caracter a textului și identificarea numărului de perechi de caractere alăturate care pot reprezintă finalul unui cuvânt (caracter diferit de spațiu, urmat de un separator).

<pre> 1 var s:string; 2 i,nr:integer; 3 begin 4 readln(s); 5 s:=s+' '; nr:=0; 6 for i:=1 to length(s)-1 do 7 if(s[i]<>' ')and 8 (s[i+1]=' ')then inc(nr); 9 writeln(nr); 10 end. 11 </pre>	<pre> #include <stdio.h> #include <string.h> char s[256]; int nr,i; void main() { gets(s); strcat(s," "); nr=0; for (i=0;i<strlen(s);i++) if (s[i]!=' ' && s[i+1]==' ') nr++; printf("%d\n",nr); } </pre>
--	--

2. Se citește de la tastatură un vers al unei poezii și o silabă. Să se realizeze un program care determină numărul de apariții al silabei citite în textul respectiv.

Exemplu : Pentru versul 'Un curcubeu multicolor' și silaba 'cu' se va afișa 2.

Soluție

Atât versul citit cât și silaba vor fi reținute în variabile de tip șir de caractere *vers* și *s*. Algoritmul propus se bazează pe căutarea repetată a subșirului *s*, folosindu-ne de funcția predefinită *pos()* / *strstr()*.

<pre> 1 var vers,s:string; 2 nr,p,l:integer; 3 begin 4 readln(vers); 5 readln(s); 6 nr:=0; 7 while pos(s,vers)<>0 do begin 8 p:=pos(s,vers); 9 l:=length(s); 10 delete(vers,p,l); 11 inc(nr); {nr:=nr+1} 12 end; 13 writeln(nr); 14 end. </pre>	<pre> #include <stdio.h> #include <string.h> char vers[256],s[256]; int nr,p,l; void main() { gets(vers); gets(s); nr=0; while (strstr(vers,s)!=NULL){ p=strstr(vers,s)-vers; l=strlen(s); strcpy(vers+p,vers+p+l); nr++; } printf("%d\n",nr); } </pre>
---	---

3. O propoziție se consideră fiind palindrom dacă ignorând diferențele dintre minuscule și majuscule și ignorând separatorii, va fi identică cu propoziția obținută prin citirea literelor de la dreapta spre stânga. De exemplu, propoziția 'Ele fac cafele' este palindrom. Să se realizeze un program care permite citirea propoziției de la tastatură și verifică dacă ea poate fi considerată palindrom. Cuvintele vor fi separate în cadrul propoziției prin spații (singurul separator prezent).

Soluție:

Algoritmul propus conține trei etape:

- ștergerea spațiilor dintre cuvinte;
- transformarea în majuscule a fiecărei litere;
- cuvântul astfel obținut se verifică dacă este palindrom.

<pre> 1 var s:string; ok:boolean; 2 i,p:integer; 3 begin 4 readln(s); 5 while pos(' ',s)<>0 do begin 6 p:=pos(' ',s); 7 delete(s,p,1); 8 end; 9 for i:=1 to length(s) do 10 s[i]:=upcase(s[i]); 11 ok:=true; </pre>	<pre> #include <string.h> #include <stdio.h> char s[256]; int i,p,ok; void main() { gets(s); while (strstr(s," ")!=NULL) { p=strstr(s," ")-s; strcpy(s+p,s+p+1);} for (i=0;i<strlen(s);i++) if (s[i]>='a' && s[i]<='z') s[i]='A'-'a'; </pre>
---	---

```

12 for i:=1 to length(s) div 2 do
13   if s[i]<>s[length(s)-i+1]
14   then
15     ok:=false;
15   writeln(ok);
16 end.
17
ok:=1;
for(i:=0;i<strlen(s)/2;i++)
  if (s[i]!=s[strlen(s)-i-1])
    ok=0;
printf("%d\n",ok);
}

```

4. Se citește de la tastatură un șir de caractere care reprezintă un număr în baza 16. Să se realizeze un program care permite conversia în baza 10 a numărului citit. În situația în care șirul conține caractere nepermise se va afișa mesajul "Imposibil".
Exemplu: Pentru numărul hexazecimal A1B se va afișa 2587.

Soluție

Știm că singurele cifre permise la scrierea unui număr într-o bază b ($2 \leq b \leq 10$) sunt $0..b-1$, corespunzătoare resturilor care se pot obține la împărțirea cu b . Dacă baza este mai mare ca 10, atunci fiecare din resturile obținute vor fi codificate în ordine cu literele A,B,C, ș.a.m.d. În baza 16 resturile mai mari ca 9 sunt codificate astfel: 'A'=10; 'B'=11; 'C'=12; 'D'=13; 'E'=14; 'F'=15.

De exemplu, numărul $2AC_{(16)}$ reprezintă numărul $2 \cdot 16^2 + 10 \cdot 16^1 + 12 \cdot 16^0 = 684_{(10)}$; Pentru a evita calcularea puterii lui 16 din cadrul fiecărui termen putem rescrie expresia de mai sus și sub forma: $2 \cdot 16^2 + 10 \cdot 16^1 + 12 \cdot 16^0 = ((0 \cdot 16 + 2) \cdot 16 + 10) \cdot 16 + 12$.

În cazul în care cifrele sunt exprimate prin litere se va realiza o corespondență între codul ASCII al său și numărul pe care îl reprezintă în baza 16:

Caracter	Codul ASCII	Restul pe care îl indică în baza 16
'A'	65	10 = 65 - 55
'B'	66	11 = 66 - 55
'C'	67	12 = 67 - 55
'D'	68	13 = 68 - 55
'E'	69	14 = 69 - 55
'F'	70	15 = 70 - 55
		ord(Litera) - 55

```

1 var s:string;
2   i,nr,c,er:integer;
3 begin
4   readln(s); nr:=0;
5   for i:=1 to length(s) do
6   begin
7     if s[i]<='9' then
8       val(s[i],c,er)
9     else
10      c:=ord(upcase(s[i]))-55;
11      nr:=nr*16 + c;
12    end;
13    writeln(nr);
14  end.

```

```

#include <string.h>
#include <stdio.h>
char s[256];
int i,nr,c;
void main() {
  gets(s); nr=0;
  for (i=0;i<strlen(s);i++) {
    if (s[i]<='9') c=s[i]-'0';
    else c=s[i]-55;
    nr=nr*16+c;
  }
  printf("%d\n",nr);
}

```

5. Se citește de la tastatură un vers al unei poezii. Să se realizeze un program care determină numărul de cuvinte din text. Cuvintele sunt separate între ele prin caracterele: spațiu (' '), virgula (,), punctul (.) sau punct și virgula (;).
Exemplu: Pentru propoziția 'Vremea trece... vremea vine,' se va afișa valoarea 4.

Soluție

În cadrul algoritmul pe care îl prezentăm vom număra câte cuvinte sunt în vers folosindu-ne de următoarea regulă:

O poziție în cadrul versului reprezintă începutul unui nou cuvânt dacă tipul caracterului de pe poziția respectivă este literă iar cel precedent este un separator. Pentru a identifica mai ușor tipul unui caracter vom folosi o variabilă șir de caractere a cărei valoare o va reprezenta șirul de separatori. '.,;'. Dacă un caracter al versului nu se regăsește printre caracterele acestuia atunci el este de tip literă. În C++ se poate folosi funcția `strtok()`.

```

1 var s,sep:string;
2   x,y,nr,i:integer;
3 begin
4   readln(s);
5   nr:=0;
6   sep:='.,;';
7   s:= '.' + s + '.';
8   for i:=2 to length(s) do
9   begin
10    x:=pos(s[i],sep);
11    y:=pos(s[i-1],sep);
12    if (x=0)and(y<>0) then
13      nr:=nr+1;
14    end;
15    writeln(nr);
16  end.

```

```

#include <string.h>
#include <stdio.h>
char s[256],*p;
int nr;
void main()
{
  gets(s);
  nr=0;
  p=strtok(s,".,;");
  while (p!=NULL) {
    nr++;
    p=strtok(NULL,".,;");
  }
  printf("%d\n",nr);
}

```

6. Se consideră două cuvinte formate din literele mari și mici ale alfabetului englez. Verificați dacă ele sunt *anagrame*.

Două șiruri de caractere sunt anagrame, dacă unul dintre ele este format din caracterele celui alt, eventual într-o altă ordine. Exemplu: 'are', 'era'.

Soluție:

Algoritmul propus caută succesiv prima literă a primului cuvânt citit (x), în cel de al doilea (y). În cazul în care este găsită ea va fi ștearsă din ambele cuvinte. Procedeu continuă până când fie litera nu este regăsită, fie când lungimea lui x este 0, caz în care cuvintele sunt anagrame.

O altă metodă ar consta în ordonarea caracterelor ambelor cuvinte și compararea acestora la final.


```

1 var x,y:string;
2 begin
3   readln(x);
4   readln(y);
5   if length(x)=length(y) then
6     begin
7       while pos(x[1],y)<=0 do
8         begin
9           delete(y,pos(x[1],y),1);
10          delete(x,1,1);
11        end;
12      if (x='') and (y='') then
13        write('Da')
14      else
15        write('Nu')
16      end
17    else write('Nu')
18  end.

```

```

#include <iostream.h>
#include <string.h>
char x[21],y[21],*p;
void main() {
  cin>>x>>y;
  if (strlen(x)==strlen(y)) {
    while (strchr(y,x[0])!=NULL
      && x[0]!=0){
      p=strchr(y,x[0]);
      strcpy(p,p+1);
      strcpy(x,x+1);
    }
    if (x[0]==0 && y[0]==0)
      cout<<"Da";
    else cout<<"Nu";
  }
  else cout<<"Nu";
}

```

7. De la tastatură se citește un text codificat după regula următoare: în fața fiecărui caracter este scris numărul de apariții consecutive ale acestuia. Realizați un program care decodifică textul. Numărul de apariții consecutive ale unui caracter este strict mai mic decât 10.

Exemplu: Pentru codificarea '1c1o1p3i' se va afișa textul 'copiii'

Soluție

În funcție de tipul fiecărui caracter (literă/cifră) se decide dacă trebuie realizată afișarea caracterului curent sau actualizarea cifrei care va reprezenta numărul de scrieri ale caracterului următor.

```

1 var s:string;
2   i,j,er,cifra:integer;
3 begin
4   readln(s);
5   for i:=1 to length(s) do
6     begin
7       if s[i] in ['0'..'9'] then
8         val(s[i],cifra,er)
9       else
10        for j:=1 to cifra do
11          write(s[i]);
12        end;
13      end.

```

```

#include <stdio.h>
#include <string.h>
char s[256];
int i,j,cifra;
void main() {
  gets(s);
  for (i=0;i<strlen(s);i++)
    if (s[i]>='0' && s[i]<='9')
      cifra=s[i]-'0';
    else
      for (j=0;j<cifra;j++)
        printf("%c",s[i]);
  printf("%c",s[i]);
}

```

8. Se citește de la tastatură un șir de caractere. Identificați în cadrul acestuia o secvență de lungime maximă care poate fi convertită către o variabilă de tip întreg.

Exemplu: Pentru șirul „25AB32042Xs23” se va afișa 32042.

Soluție

Algoritmul propus determină, după o singură parcurgere a caracterelor din șir, care este secvența de lungime maximă ce poate fi convertită către o dată de tip numeric. Se va reține în final, poziția de început (*poz*) a secvenței în cadrul șirului și lungimea acesteia (*max*).

```

1 var s,c:string;
2   i,lung,max,poz,er,x:integer;
3 begin
4   readln(s); lung:=0;
5   s:=s+' ';
6   for i:=1 to length(s) do
7     begin
8       val(s[i],x,er);
9       if (er=0) then inc(lung)
10      else begin
11        if max<lung then begin
12          max:=lung;
13          poz:=i-lung;
14        end;
15        lung:=0;
16      end;
17    end;
18  for i:=poz to poz+max-1 do
19    write(s[i]);
20 end.

```

```

#include <stdio.h>
#include <string.h>
char s[256];
int i,lung,max,poz;
void main() {
  gets(s); strcat(s," ");
  lung=0;
  for (i=0;i<strlen(s);i++)
    if (s[i]!='0' && s[i]<='9')
      lung++;
    else {
      if (max<lung) {
        max:=lung;
        poz:=i-lung;
      }
      lung=0;
    }
  for (i=poz;i<poz+max;i++)
    printf("%c",s[i]);
}

```

9. Se dorește ca operația *Find-Replace* să fie executată asupra unui text care nu conține mai mult de 250 de caractere. Această operație constă în înlocuirea tuturor aparițiilor unui subșir *s1* cu un alt subșir *s2*. În cazul de față cele două subșiruri se consideră a fi diferite și de lungimi egale. Creați un program care simulează această operație.

Exemplu: Considerând textul „care caracatita”, dacă subșirul „ca” se va înlocui cu „ta” atunci textul afișat va fi „tare taratatita”.

Soluție

Programul sursă realizat pentru varianta Pascal folosește apeluri la subprogramele predefinite pentru tipul șir de caractere. Sursa C/C++ construiește un nou șir de caractere *S* în care subșirul *s1* a fost înlocuit cu *s2*.

```

1 var
2   s,s1,s2:string;
3   c:integer;
4
5 begin
6   readln(s);
7   readln(s1);
8   readln(s2);
9   c:=pos(s1,s);

```

```

#include <stdio.h>
#include <string.h>
char S[256],s1[56];
int u,i,p;
void main() {
  gets(s);
  gets(s1);
  gets(s2);
}

```

```

10 while c<>0 do
11 begin
12   delete(s,c,length(s1));
13   insert(s2,s,c);
14   c:=pos(s1,s);
15 end;
16 writeln(s);
17 end.
18
19 do ( char* pt=strstr(s,s1);
20   if (!pt){
21     for(i=u;i<strlen(s);i++)
22       strncat(S,s+i,1); break;
23   } p=pt-s;
24   for(i=u;i<p;i++) S[i]=s[i];
25   for(i=p;i<p+strlen(s1);i++)
26     s[i]='!';
27   for(i=p;i<p+strlen(s2);i++)
28     strncat(S,s2+i-p,1);
29   u=p+strlen(s1); while (1);
30   puts(S);
31 }

```

10. Se consideră un șir de n cuvinte. Identificați mulțimea cu număr maxim de cuvinte care sunt anagrame între ele două câte două. Se va afișa cardinalul mulțimii și un element al acesteia, considerat reprezentantul ei.

Exemplu: pentru $n=6$ și cuvintele 'arc', 'rac', 'voi', 'car', 'armata', 'tamara' se va afișa mulțimea: arc 3.

Soluție

Algoritmul verifică pentru oricare două cuvinte dacă sunt anagrame. Se memorează elementul pentru care s-a determinat numărul maxim de anagrame.

```

1 var a:array[1..100]of
2   string[20];
3   n,i,nr,j,max:integer;
4   cuv,x,y:string;
5 begin
6   readln(n); max:=0;
7   for i:=1 to n do
8     readln(a[i]);
9   for i:=1 to n-1 do begin
10    nr:=1;
11    for j:=i+1 to n do begin
12      y:=a[j]; x:=a[i];
13      if length(x)=length(y) then
14        begin
15          while pos(x[1],y)<>0 do
16            begin
17              delete(y,pos(x[1],y),1);
18              delete(x,1,1);
19            end;
20          if (x='')and(y='')then
21            inc(nr);
22          end;
23          if max<nr then begin
24            max:=nr; cuv:=a[i];
25          end;
26        end;
27      end;
28      writeln(cuv,' ',max);
29    end.

```

```

#include <iostream.h>
#include <string.h>
char a[101][21],x[21],y[21];
char *p ,cuv[21];
int nr,n,i,j,k,max,poz;
void main() {
  cin>>n; max=0;
  for (i=0;i<n;i++){
    for (j=i+1;j<n;j++){
      strcpy(x,a[i]);
      strcpy(y,a[j]);
      if (strlen(x)==strlen(y)) {
        while(strchr(y,x[0])!=NULL
          && x[0]!=0){
          p=strchr(y,x[0]);
          strcpy(p,p+1);
          strcpy(x,x+1);
        }
        if (x[0]==0 && y[0]==0)
          nr++;
      }
      if (max<nr){
        max=nr;strcpy(cuv,a[i]);
      }
    }
  }
  cout<<cuv<<" "<<max;
}

```

11. Se consideră un fișier text *in.txt* ce conține numere întregi dispuse pe mai multe linii. Orice caracter ce nu reprezintă un caracter numeric este considerat separator. Scrieți un program care creează un fișier *out.txt* ce conține pe fiecare linie media aritmetică a numerelor situate pe aceeași linie în fișierul *in.txt*. Media aritmetică va fi scrisă cu două zecimale. Pe fiecare linie a fișierului de intrare se află maximum 200 de caractere.

Exemplu: in.txt	out.txt
2..3a 403bx	136.00
2..2 A,..5	3.00
1.92	46.50

Soluție:

În problema de față, separatorii nu sunt reprezentați prin spații albe. Această situație impune folosirea la citire a unei variabile de tip șir de caractere.

Pentru determinarea mediei se va parcurge șirul citit caracter cu caracter, relizându-se conversia secvențelor de caractere numerice către date de tip numeric(întreg).

```

1 var f,g:text; x,c:string;
2   i,v,s,n,er:integer;
3 begin
4   assign(f,'in.txt'); reset(f);
5   assign(g,'out.txt');rewrite(g);
6   while not eof(f) do begin
7     s:=0; n:=0; c:='';
8     readln(f,x);
9     x:=x+'.';
10    for i:=1 to length(x) do
11      if x[i] in ['0'..'9'] then
12        c:=c+x[i];
13      else
14        if c<>'' then begin
15          val(c,v,er); c:='';
16          s:=s+v;
17          inc(n);
18        end;
19        writeln(g,s/n:0:2);
20      end;
21      close(f); close(g);
22    end.

```

```

#include <stdio.h>
#include <string.h>
FILE *f,*g; char x[256];
int i,n;
float c,s;
void main() {
  f=fopen("in.txt","r");
  g=fopen("out.txt","w");
  while (!feof(f)) {
    s=0; n=0; c=0;
    if (!fgetc(x,256,f)) break;
    strcat(x,".");
    for (i=0;i<strlen(x);i++)
      if (x[i]>='0'&&x[i]<='9')
        c=c*10+x[i]-'0';
    else if (c) {
      s+=c; n++; c=0;
    }
    fprintf(g,"%0.2f\n",s/n);
  }
  fclose(f); fclose(g);
}

```

12. Se consideră un cuvânt din maximum 50 de caractere format numai din literele alfabetului englezesc. Se cere inserarea minusculei minime din punct de vedere lexicografic (aparținând cuvântului), între oricare două litere identice aflate pe poziții alăturate. La inserare nu se va face distincție între majuscule și minuscule. Dacă nu există minuscule în cuvânt atunci acesta nu va suferi nici o modificare.

Exemplu:

Pentru cuvântul "inNorat" minuscula minimă este 'a' și se va afișa „inaNorat”. Pentru șirul de caractere „boQlutton” minuscula minimă este 'b' și se va afișa „bobOlutbton”.

Soluție

Operația de inserare se efectuează simultan cu operația de parcurgere a șirului de caractere. La inserarea unui nou caracter între două litere identice, indicele curent se incrementează cu două poziții.

```
1 var s:string; m:char; i:byte;
2
3 begin
4   readln(s);
5   m:=chr(127);
6   for i:=1 to length(s) do
7     if (s[i]<m)
8       and (s[i]<>upcase(s[i]))
9     then m:=s[i];
10  i:=1;
11  if upcase(m)<>m then begin
12    while i<length(s) do
13      if upcase(s[i])=
14        upcase(s[i+1])
15      then begin
16        insert(m,s,i+1);
17        inc(i,2);
18      end
19      else
20        inc(i);
21      end;
22    writeln(s);
23  end.
```

```
#include <string.h>
#include <stdio.h>
char s[256],m,c1,c2; int i;
void main() {
  gets(s); m=127;
  for(i=0;i<strlen(s);i++){
    c1=s[i]>='A'&&s[i]<='Z'?
      s[i]+'a'-'A':s[i];
    if (m>c1) m=c1;
  }
  for (i=0;i+1<strlen(s);i++){
    c1=s[i]>='A'&&s[i]<='Z'?
      s[i]+'a'-'Z':s[i];
    c2=s[i+1]>='A'&&s[i+1]<='Z'?
      s[i+1]+'a'-'A':s[i+1];
    if (c1==c2) {
      memmove(s+i+1,s+i,
        strlen(s)-i);
      s[++i]=m;
    }
  }
  puts(s);
}
```

13. Fișierul text „P.txt” conține pe mai multe linii un algoritm reprezentat în pseudocod. Știm că singurele cuvinte cheie ce se regăsesc în fișier sunt: *intreg, daca, atunci, altfel, citește, scrie, stop*. Să se determine numărul de variabile folosite în algoritm și identificatorii acestora. Toate caracterele care intervin în fișier au coduri ASCII asociate. Liniile din fișier se termină cu caracterul punct și virgulă(,).

Exemplu: Pentru secvența „daca a>b atunci xx=c altfel d=a+b stop;” se va afișa:

```
5
a b xx c d
```

Soluție

Tabloul unidimensional *v* va memora toate variabilele identificate. Fișierul va fi parcurs caracter cu caracter, formându-se de fiecare dată o secvență ce poate reprezenta un cuvânt cheie sau identificatorul unei variabile. Dacă acesta nu reprezintă un cuvânt cheie, atunci este salvat în vectorul *v*, numai în cazul în care nu a fost deja memorat.

```
1 type
2   sir=array[1..7] of string[10];
3   const a : sir=('intreg',
4     'citește', 'scrie', 'daca',
5     'atunci', 'altfel', 'stop');
6   var f:text; i,n,m:integer;
```

```
#include <string.h>
#include <stdio.h>
const char a[8][11]={
  "intreg", "citește", "scrie",
  "daca", "atunci", "altfel",
  "stop"};
```

22

```
7 s:string; x:char; ok:boolean;
8 v=array[1..50] of string[10];
9 begin
10  assign(f,'p.txt'); reset(f);
11  m:=0;
12  while not eof(f) do begin
13    s:='';
14    while not eoln(f) do begin
15      read(f,x);
16      if x in ['a'..'z'] then
17        s:=s+x
18      else if s<>'' then begin
19        ok:=true;
20        for i:=1 to 7 do
21          if a[i]=s then ok:=false;
22        for i:=1 to m do
23          if s=v[i] then ok:=false;
24        if ok then begin
25          inc(m); v[m]:=s;
26          end; s:='';
27        end; end; readln(f); end;
28    writeln(m);
29    for i:=1 to m do
30      write(v[i], ' ');
31  end.
```

```
int i,j,m,ok; FILE *f;
char s[256],x[256],v[51][11];
void main() {
  f=fopen("p.txt","r");
  for (m=0;!feof(f);) {
    fgets(x,256,f);
    strcat(x," ");
    for(i=0;i<strlen(x);i++)
      if(x[i]>='a'&&x[i]<='z')
        strncat(s,x+i,1);
    else if (strlen(s)) {
      ok=1;
      for(j=0;j<7;j++)
        if(!strcmp(a[j],s))ok=0;
      for(j=0;j<m;j++)
        if(!strcmp(v[j],s))ok=0;
      if (ok) strcpy(v[m++],s);
      memset(s,0,sizeof(s));
    }
  }
  printf("%d\n",m);
  for(i=0;i<m;i++)
    printf("%s ",v[i]);
}
```

14. Se consideră un șir de *n* cuvinte. Se dorește afișarea lor pe verticală, respectând următoarele cerințe:

- pe fiecare coloană se află în ordine câte un cuvânt, de la primul până la ultimul;
- pe ultima linie se află primele litere ale fiecărui cuvânt;
- pe prima linie se află ultimele litere ale celor mai lungi cuvinte.

Exemplu: Pentru șirul de 4 cuvinte: 'eu', 'masa', 'noi', 'vine' se va afișa:

```
a e
s i n
u a o i
e m n v
```

Soluție

Se folosește ca auxiliar un vector *h* în care este memorată lungimea fiecărui cuvânt. Numărul de linii pe care se va face afișarea este egală cu elementul maxim din *h*. Pentru fiecare cuvânt, în cadrul unei linii, se poate afișa fie caracterul spațiu, dacă lungimea acestuia este mai mică decât valoarea variabilei *mx* (care contorizează poziția literei ce urmează a fi afișată), fie litera de pe poziția *mx*.

```
1 type
2   sir=array[1..99] of string;
3   var a:sir;
4   l,i,n,mx:integer;
```

```
#include <string.h>
#include <stdio.h>
int n,i,max,h[100];
char a[100][256];
```

1.1.3 Probleme propuse

```

5 h:array[1..99]of byte;
6 begin
7   readln(n);mx:=0;
8   for i:=1 to n do begin
9     readln(a[i]);
10    h[i]:=length(a[i]);
11    if h[i]>mx then mx:=h[i]
12  end;
13  for l:=1 to mx do begin
14    for i:=1 to n do
15      if h[i]=mx then begin
16        write(a[i][h[i]]);
17        dec(h[i])
18      end
19      else write(' ');
20    dec(mx);
21    writeln;
22  end;
23 end.

```

```

void main() {
  scanf("%d",&n); max=0;
  for(i=0;i<n;i++){
    scanf("%s",a[i]);
    h[i]=strlen(a[i])-1;
    if (max<h[i]) max=h[i];
  }
  for(;max>=0;max--){
    for (i=0;i<n;i++){
      if (h[i]==max){
        printf("%c",a[i][h[i]]);
        h[i]--;
      }
      else printf(" ");
    }
    printf("\n");
  }
}

```

15. Se citește de la tastatură un șir de maximum 255 litere mici ale alfabetului englez. Să se realizeze un program care identifică cea mai lungă secvență de caractere care se repetă de minimum 2 ori în cadrul șirului.

Exemplu: Pentru șirul de caractere: "masectrseacsacrrr" se va afișa „sec”.

Soluție

Algoritmul parcurge fiecare poziție din șir și identifică lungimea maximă a secvenței care poate începe cu caracterul respectiv. Variabila *x* reține această secvență.

```

1 var l,max,i:integer;
2 s,r,x,y:string;
3 begin
4   readln(s); max:=0;
5   for i:=1 to length(s)-1 do
6     begin
7       l:=0; x:='';
8       repeat
9         inc(l); y:=s;
10        x:=x+s[i+l];
11        delete(y,i,length(x));
12      until (pos(x,y)=0)or
13        (l+i>length(s));
14      if length(x)-1>max then begin
15        max:=length(x)-1;
16        r:=copy(x,1,max);
17      end;
18    end;
19    writeln(r);
20  end.
21

```

```

#include <string.h>
#include <stdio.h>
int l,max,i; char
s[256],r[256],x[256],y[256];
void main() {
  gets(s); max=0;
  for(i=0;i+1<strlen(s);i++){
    l=0; memset(x,0,sizeof(x));
    do {
      strncat(x,s+i+1,1);l++;
      strcpy(y,s);
      strcpy(y+i,y+i+strlen(x));
    } while (i+1<strlen(s) &&
      strstr(y,x));
    if (max<strlen(x)-1){
      max=strlen(x)-1;
      strncpy(r,x,max);
    }
  }
  puts(r);
}

```

1. Se consideră un text de maximum 255 de caractere. Realizați un program care afișează:

- Numărul de apariții al unei litere în text. Litera va fi citită de la tastatură.
- Câte vocale apar în textul citit.
- Numărul de apariții al unei silabe în text. Silaba va fi citită de la tastatură.

Exemplu:

Pentru litera 'a', silaba „re” și textul:
„Îna are multe mere”

se va afișa:

- 2
- 8
- 2

2. Se consideră un cuvânt format din litere mici și mari ale alfabetului englez. Realizați un program care permite ștergerea tuturor aparițiilor primei litere în cadrul cuvântului respectiv.

Exemplu: Pentru cuvântul 'mamaie' se va afișa 'aaie'.

3. Se consideră un cuvânt format din literele mici și mari ale alfabetului englez. Să se scrie un program care afișează cuvintele obținute din cuvântul inițial prin eliminarea succesivă a primului și ultimului caracter al șirului.

Exemplu: Pentru cuvântul 'Deosebit' se va afișa:

eosebi
oseb
se

4. Se consideră o matrice de dimensiune *n*×*m* cu elemente de tip șir de caractere. Creați un program care afișează șirul de caractere de lungime maximă de pe fiecare linie a matricei.

5. Să se creeze un program care transformă literele mici ale unui cuvânt în litere mari și literele mari în litere mici.

Exemplu: Pentru cuvântul 'MiorItIC' se va afișa: 'mIORiTic'.

6. Se consideră un șir de *n* cuvinte. Să se determine cuvântul de lungime maximă care se poate forma prin concatenarea a două dintre cuvintele citite.

Exemplu: Pentru *n*=5 și șirul de cuvinte: 'mama', 'arc', 'conduce', 'paine', 'vine' se va afișa: 'conducepaine' sau 'paineconduce'.

7. Se consideră un șir de *n* cuvinte. Să se determine cuvântul cel mai mic în ordine lexicografică obținut prin concatenarea a două dintre cuvintele citite.

Exemplu: Pentru *n*=5 și șirul: 'mama', 'arc', 'conduce', 'paine', 'vine' se va afișa: 'arconduce'.

8. Considerăm un text de maximum 255 de caractere. Propozițiile sunt delimitate prin caracterele punct(.) sau prin semnul exclamării(!). Realizați un program care afișează fiecare propoziție pe o singură linie, fiecare cuvânt începând cu o majusculă.

Exemplu: Pentru textul 'Fie A un punct fix.Presupunem B punct mobil.' se va afișa:

Fie A Un Punct Fix.

Presupunem B Punct Mobil.

9. Se consideră o listă cu n ($n < 100$) prenume ale elevilor dintr-o clasă. Prenumele unei fete este recunoscut datorită faptului că fie are ultima literă 'a', fie este 'Carmen' sau 'Alice'. Să se creeze un program care afișează numărul fetelor din clasă și cel mai mare prenume în sens lexicografic ale cărui litere vor fi transformate în majuscule.

Exemplu: $n=5$ și lista 'Ana', 'Alice', 'Mihai', 'Maria', 'Ion', se va afișa: '3 MIHAI'.

10. Se consideră 2 cuvinte ce conțin numai litere mici. Considerăm că ultima silabă a unui cuvânt este subșirul care începe cu ultima lui vocală. Verificați dacă aceste cuvinte rimează (dacă au ultima silabă identică). Dacă un cuvânt nu conține vocale, atunci ultima silabă este întregul cuvânt.

Exemplu: Pentru cuvintele 'armat' și 'verificat' se va afișa mesajul 'Rimeaza'.

11. De la tastatură se citește un text codificat după regula următoare: în fața fiecărui caracter este scris un număr ce reprezintă numărul de apariții consecutive al acestuia. Realizați un program care decodifică textul. Numărul ce apare în fața unui caracter va fi mai mic sau cel mult egal cu 20.

Exemplu: Pentru '1G11o1L' se va afișa 'GoooooooooL'.

12. Se consideră un text în care spațiul este unicul separator. Realizați un program care afișează numerele ce apar în text, despărțite prin câte un spațiu.

Exemplu: Pentru textul 'Ana are 7 mere si 223 de pere' se va afișa: '7 223'.

13. Se citește de la tastatură un șir de caractere. Realizați un program care determină cea mai lungă secvență de cifre alăturate din șir.

Exemplu: Pentru șirul de caractere 'a23Bw001234mcy34' se va afișa '001234'.

14. Se dorește ca operația *Find-Replace* să fie executată asupra unui text care nu conține mai mult de 255 de caractere. Această operație constă în înlocuirea tuturor aparițiilor unui subșir $s1$ cu un alt subșir $s2$. Realizați un program care simulează această operație. Șirurile $s1$ și $s2$ nu au neapărat aceeași lungime.

Exemplu: Pentru textul 'are mere și pere' $s1='re'$ și $s2='rare'$ se va afișa: 'arare merare si perare'.

15. Se consideră un șir de n cuvinte. Realizați un program care identifică toate anagramele primului cuvânt care se regăsesc în șir.

Exemplu: pentru $n=5$, cuvintele 'arc', 'rac', 'eu', 'tu', 'car' se va afișa: 'rac' și 'car'

16. Se consideră o propoziție care are maximum 255 de caractere. Orice caracter diferit de literă este considerat separator. Realizați un program care afișează fiecare cuvânt pe o linie a ieșirii standard.

Exemplu: „Am venit!...” se va afișa:

Am

venit

17. Se consideră un text de maximum 255 de caractere. Realizați un program care determină cea mai lungă secvență de litere alăturate care apar în ordine alfabetică.

Exemplu: 'AMC WCDRVAS' se va afișa 'CDRV'.

18. Se consideră un text de maximum 255 de caractere. Realizați un program care determină cea mai lungă secvență de litere care reprezintă un palindrom.

Exemplu: 'AM tCOjOCn wqaqd' se va afișa 'COjOC'.

19. Se consideră un text de maximum 255 de caractere litere mici sau spații. Realizați un program care rescrie textul astfel încât cuvintele să apară ordonate alfabetic.

Exemplu: Pentru șirul de caractere 'ieri am venit devreme' se va afișa:

'am devreme ieri venit'.

20. Se consideră un text de maximum 255 de caractere. Realizați un program care inversează literele fiecărui cuvânt:

Exemplu: Pentru șirul de caractere 'Am venit!...spune el' se afișează:

'mA tinev !...enups le'.

21. Fie un șir cu maximum 100 de caractere alfanumerice. Să se afișeze toate subșirurile formate din două caractere alăturate care pot reprezenta un număr natural de două cifre, separate prin câte un singur spațiu.

Exemplu: Pentru șirul 'w234b5br6779' se va afișa: '23 34 67 77 79'.

22. De la tastatură se citește un șir de maximum 10 caractere. Realizați un program care verifică dacă acest șir poate reprezenta un număr real exprimat în forma zecimală cu 3 zecimale exacte.

Exemplu: Pentru șirul „31.0.234” se va afișa mesajul 'NU'.

23. Creați un program care afișează numărul vocalelor dintr-o dată de tip șir de caractere împreună cu șirul obținut după ștergerea lor.

Exemplu: Pentru șirul de caractere: "moale" se va afișa: '3 ml'.

24. Se consideră o propoziție care are maximum 250 de caractere. Ea cuprinde cuvinte formate din literele alfabetului englezesc. Cuvintele sunt despărțite prin unul sau mai mulți separatori. Orice caracter care nu este literă va fi considerat separator. Creați un program ce va afișa pe ecran fiecare cuvânt din propoziție pe câte o linie. Cuvintele vor avea prima literă majusculă.

Exemplu: Pentru propoziția:

'-Ai venit? intreba Alina' se va afișa pe ecran:

Ai

Venit

Intreba

Alina

25. Se citește de la intrarea standard un șir de maximum 200 de caractere reprezentând o propoziție. Cuvintele sunt secvențe de litere mici sau mari ale alfabetului englezesc. Realizați un program care afișează propoziția după ce cuvintele au fost ordonate lexicografic. După fiecare cuvânt vor fi afișați separatorii prezenți în propoziția inițială după cuvântul cu numărul de ordine respectiv.

Exemplu: Pentru textul „Ce spui.....?”, a întrebat el.' se va afișa:

'Ce a.....?, el întrebat spui'.

26. Se consideră un șir de cel mult 100 de caractere, citit de pe prima linie a fișierului *in.txt*. Să se afișeze toate șirurile de caractere de lungime maximă $2*k$ obținute prin concatenarea prefixelor cu sufixele de aceeași lungime ale acestui șir. Valoarea lui k se citește de la tastatură.

Exemplu:

Pentru $k=4$

și fișierul *in.txt*

caracatita.

se va afișa:

ca

cata

carita

caratita

27. Fișierul *in.txt* conține un text dispus pe mai multe linii. Orice caracter care nu reprezintă o literă mică a alfabetului englezesc se consideră separator. Numărul de caractere ale unei linii nu depășește 200. Realizați un program care afișează numărul de apariții ale unui cuvânt; preluat de la tastatură, în textul din fișier.

Exemplu: Pentru cuvântul „venit”

și fișierul *in.txt*

Am venit acasa!

EA a, l venit!

el a venit acum**?!

se va afișa: 3

28. Conținutul fișierului *virus.txt* a fost deteriorat, iar valorile naturale care inițial erau separate în cadrul liniilor prin câte un singur spațiu, au acum ca separatori orice caracter ce nu reprezintă o cifră. Realizați un program care creează fișierul *sum.txt* în care pe fiecare linie se află suma numerelor aflate în fișierul *virus.txt*.

Exemplu: Pentru fișierul *virus.txt*

w23, ,mmmm230...20e

fsdj.4.sal..45,,sk56ddd90z

fișierul *sum.txt* va conține:

273

195

29. Se consideră o propoziție de maximum 50 de caractere, citită de la tastatură. Să se realizeze un program care afișează propoziția după eliminarea tuturor cuvintelor de lungime p .

Exemplu: Pentru $p=3$ și propoziția „Noi am gandit la fel ca voi...!” se va afișa: „am gandit la ca ...!”

30. Fie un cuvânt de maximum 50 de caractere litere mici ale alfabetului englez. Să se afișeze toate cuvintele distincte obținute prin eliminarea unor secvențe de p litere aflate pe poziții consecutive în cuvântul inițial. Cuvintele afișate vor fi despărțite prin câte un spațiu.

Exemplu: Pentru cuvântul „mamaie” și $p=2$ se va afișa: 'maie mama mame'.

31. Se citesc de la tastatură două cuvinte formate doar din literele minuscule ale alfabetului englezesc. Se cere realizarea unui program care afișează:

- mulțimea literelor comune celor două cuvinte (intersecția);
- mulțimea literelor care se găsesc în unul din cele două cuvinte (reuniunea);
- mulțimea literelor care se găsesc în primul cuvânt citit și nu apar în al doilea.

Exemplu: Pentru cuvântul „mamaie” și cuvântul „macara” se va afișa:

{m,a}

{m,a,i,e,r,c}

{i,e}

32. Fișierul *in.txt* conține un text, fiecare propoziție fiind dispusă pe o singură linie. Afișați pe ecran propoziția de lungime maximă, obținută după eliminarea tuturor secvențelor de două caractere consecutive, ambele vocale.

Exemplu: Pentru propoziția:

Mama_Are_Mere

Maine_este_o_noua_zi

se va afișa:

Mne_este_o_na_zi

33. Se citește de la tastatură o propoziție de maximum 200 de caractere. Afișați pe ecran, pe mai multe linii, propoziția trunchiată astfel: pe ultima linie va fi afișat ultimul caracter al propoziției, pe penultima, următoarele ultime două caractere (antepenultimul și penultimul), ș.a.m.d. Pe prima linie se vor găsi primele n caractere ale propoziției, unde n este mai mic sau egal cu numărul de linii pe care s-a făcut afișarea.

Exemplu: Pentru propoziția:

abcdefghijkl

se va afișa:

ab

cdef

ghi

jk

l

34. Să considerăm următorul șir:

a, b, ba, bba, bbaba, babbaba, babbababbaba

Scrieți un program care să determine care este cel de-al n -lea termen al șirului. Valoarea lui n este strict mai mică decât 20.

Exemplu: Pentru $n=6$ se va afișa: 'babbaba'.

35. Se consideră un șir de n cuvinte reprezentând numele disciplinelor din orarul zilei următoare ale unui elev. Realizați un program care permite afișarea orarului pe ieșirea standard, astfel încât materiile să fie scrise pe coloane, în ordine lexicografică.

Exemplu: Pentru $n=4$ și disciplinele „muzica”, „desen”, „sport”, „biologie”, se va afișa sub forma:

```
b d m s
i e u p
o s z o
l e i r
o n c t
g a
i
e
```

36. Se consideră un text ce conține maximum 100 de cuvinte, fiecare cuvânt fiind format din cel mult 50 de litere ale alfabetului englez. Orice alt caracter este considerat separator. Textul este dispus pe mai multe linii în fișierul *in.txt*. Să se afișeze pe o singură linie a ieșirii standard, cuvintele de lungime maximă din text, separate prin câte un spațiu.

Exemplu: *in.txt*

```
Am venit . . . acasa
maine , plec!
Tu vrei sa pleci****?
```

se va afișa:
venit acasa maine pleci

37. În fișierul *in.txt* se află dispuse pe mai multe linii informații despre ora de plecare a trenurilor și destinația lor finală. Momentul plecării din stație este exprimat prin 5 caractere, primele două reprezentând ora, iar ultimele două reprezentând minutele(*hh:mm*). În continuare, în cadrul liniei se află numele destinației. Afișați, pe câte o linie a ieșirii standard, lista destinațiilor în ordinea crescătoare a momentului plecării. Pe fiecare linie se va afișa numele destinației și ora plecării, separate printr-un spațiu. Ora de plecare va fi exprimată în minute.

Exemplu: *in.txt*

```
10:23Cluj
15:04Arađ
05:00Iasi
```

se va afișa:
Iasi 300
Cluj 623
Arađ 904

38. Considerăm validă o adresă de e-mail dacă este formată doar din litere mici ale alfabetului englez, din caracterul '.' și caracterul '@' care apare o singură dată. Caracterele alăturate acestuia nu pot fi puncte '.'. Se codifică o adresă de e-mail după următoarele reguli:

- caracterul '@' este înlocuit prin secvența 'at' numai în cazul în care nu este precedat sau urmat de caracterul punct.
- secvența de caractere 'cod' poate fi introdusă oriunde în adresă, dar nu mai mult de o dată.

Dându-se o adresă codată, afișați pe câte o linie adresele de e-mail din care ar fi putut proveni.

Exemplu:

Pentru adresa:

sco.at.matatam.icodm

se va afișa:

```
sco.at.m@atam.icodm
sco.at.m@atam.im
sco.at.mat@am.icodm
sco.at.mat@am.im
```

39. O formulă este restrânsă folosind parantezele rotunde astfel: șirul „ababab” va fi scris simplificat „(ab)3” iar șirul „axzyxzyw” va fi scris „a(xzy)2w”. Formula restrânsă se citește de la intrarea standard. Știind că nu sunt prezente perechi de paranteze incluse una în alta, afișați șirul inițial.

Exemplu: Pentru șirul s(ad)3f(ser)2, se va afișa „sadayadfserser”.

40. Se citește, de la intrarea standard, un șir de maximum 200 de caractere. Realizați un program care determină cea mai lungă secvență de caractere ordonate lexicografic. În situația în care există mai multe astfel de secvențe se vor afișa toate, fiecare pe câte o linie a ieșirii standard.

Exemplu: Pentru textul „ABCADEMIA” se va afișa

ABC
AED

41. Se citește de la intrarea standard un șir de maximum 200 de caractere. Realizați un program care determină cea mai lungă secvență de caractere cu proprietate palindromică. În situația în care există mai multe astfel de secvențe se vor afișa toate, fiecare pe câte o linie a ieșirii standard.

Exemplu: Pentru textul „Acojoc tar eabbbar” se va afișa

cojoc
abbba

42. Se consideră o expresie aritmetică ce conține numai operatorii {+,*}. Operanzii sunt reprezentați prin variabile ai căror identificatori sunt descriși printr-o singură literă, sau prin numere naturale. Realizați un program care verifică dacă expresia este corectă sintactic.

Exemplu: Pentru expresia 23*a+ba*+3 se va afișa „Incorect”.

Pentru expresia 23*a+b*3 se va afișa „Corect”.

43. Se consideră un șir de paranteze rotunde care intervin într-o expresie aritmetică. Realizați un program care verifică dacă acestea formează un șir de paranteze care se închid corect(după regulile aritmetice).

Exemplu:

Pentru șirul de paranteze „(() ())” se afișează mesajul „Corect”.

Pentru șirul de paranteze „()) () ()” se afișează mesajul „Incorect”.

44. Se consideră următoarea regulă de codificare a unui cuvânt: în fața oricărei vocale a cuvântului se inserează caracterul ‘p’. Realizați un program care realizează operația de codificare și de decodificare a unui cuvânt. De la tastatură se va introduce tipul operației dorite prin caracterele: *C* pentru codificare și *D* pentru decodificare.

Exemplu:

Pentru operația *C* și cuvântul „oaie” se va afișa: „popapipe”.

Pentru operația *D* și cuvântul „pparpintpe” se va afișa: „parinte”.

45. Se consideră un număr rațional n ($n \leq 30000$) care are partea fracționară formată din maximum 6 cifre. Să se scrie un program care efectuează conversia numărului n în baza 2. Datele se citesc de la tastatură și se afișează pe ecran.

Exemplu: Pentru $n=7.375$ se va afișa:

„111.011”.

46. În fișierul text *base.in* sunt dispuse pe fiecare linie câte un număr. În scrierea lor intervin cifrele de la 0..9 și literele de la A la M (reprezentând, în ordine, valorile 10, 11, 12, ...). Baza în care sunt reprezentate acestea se consideră a fi baza minimă comună tuturor. Realizați un program care determină intervalul $[a,b]$, de lungime minimă, în care sunt incluse toate numerele din fișier, în urma conversiei în baza 10.

Exemplu:

Pentru fișierul *base.in*

07B

1AA

C8A

se va afișa:

a=102

b=2142

47. Se consideră o secvență de n șiruri de caractere, fiecare reprezentând o parolă formată din maximum 20 de caractere (cifre 0..9 sau majuscule). Din fiecare parolă se elimină succesiv caracterele de la stânga spre dreapta până la cel mai mare caracter al său în sens lexicografic. De exemplu, parola „A250B089” devine „B089”. În urma acestei operații, unele dintre parole devin identice și sunt considerate invalide. Ele sunt înlocuite cu suma dintre poziția ei în șir și suma codurilor ASCII a caracterelor primei parole valide rămase în șir. Operația devine imposibilă dacă și noua parolă obținută este invalidă. Realizați un program care afișează șirul parolilor după aceste transformări sau mesajul „Imposibil”.

Exemplu: Pentru $n=3$ și șirurile

07BA023

AABBAB

A07BA023

se va afișa:

281

BBAB

283

1.2. Tipul înregistrare – structură

1.2.1 Teste cu alegere multiplă și duală

1. Care dintre următoarele afirmații sunt adevărate?

- a) O înregistrare nu poate avea două câmpuri desemnate prin același identificator;
- b) O înregistrare nu poate avea două câmpuri din același tip;
- c) Orice înregistrare are cel puțin un câmp;
- d) Numărul de câmpuri al unei înregistrări nu poate depăși 20.

2. Care dintre următoarele definiții de tip sunt corecte sintactic?

a) `type A : record
 c1:byte; c2:byte;
end;`

b) `type A = record
 c1, c2 :byte;`

c) `type A = record;
 c1:boolean;
 c2:byte; end;`

d) `type A = record
 c1:boolean; c2:byte
end;`

a) `typedef A struct {
 int c1,c2;
};`

b) `typedef A struct
 int c1,c2;`

c) `typedef struct A; {
 int c1,c2;
};`

d) `typedef struct {
 int c1,c2;
} A;`

3. Considerând următoarea declarație, specificați care dintre referirile de mai jos nu reprezintă un caracter:

`Type Exemplu = record
 a:real; b:string[10];
 c:array[1..10] of char
end;
Var x, y: exemplu;`

`typedef struct {
 float a;
 char *b, c[10];
} Exemplu;
Exemplu x, y;`

a) `x.b`

b) `y.b[1]`

c) `x.c[2]`

d) `y.a`

4. Presupunem că următoarele declarații au fost făcute pentru a reține numele și vârsta elevilor dintr-o clasă, în ordine alfabetică. Specificați care dintre referirile de mai jos reprezintă inițiala numelui primului elev din catalog:

`Type elev=record
 nume: array [0..255] of char;
 varsta: integer`

`typedef struct {
 char nume[256];
 int varsta;`


```
end;
clasa = array[0..34] of elev;
Var a: clasa;
```

```
) elev;
typedef elev clasa[35];
clasa a;
```

a) a[0].elev.num
b) a[0].elev.num[0]

c) a[0].num[0]
d) clasa[0].num

5. Considerând următoarele definiții de tipuri, specificați care dintre apelurile de mai jos sunt corecte sintactic:

```
Type pers = record
  name: string;
  age: integer end;
  ap = record
    locatar: pers;
    nr: integer;
  end;
```

```
typedef struct {
  char name[256];
  int age;
} pers;
typedef struct {
  pers locatar;
  int nr;
} ap;
ap exemp;
```

a) readln(exemp.locatar).
b) readln(exemp.locatar.age).
c) readln(exemp.pers.name).
d) readln(exemp.nr).

a) cin>>exemp.locatar;
b) cin>>exemp.locatar.age;
c) cin>>exemp.pers.name;
d) cin>>exemp.nr;

6. Identificați care dintre următoarele instrucțiuni de atribuire sunt corecte sintactic, considerând declarațiile de mai jos:

```
Type pers = record
  name: string;
  age: integer end;

Var a: pers;
  b: record
    name: string;
    age: integer
  end;
```

```
typedef struct {
  char *name;
  int age;
} pers;
pers a;
struct {
  char *name;
  int age;
} b;
```

a) a.name:='';
b) a:=b;
c) a.age:=b.age;
d) a.name:=name;

a) a.name="";
b) a=b;
c) a.age=b.age;
d) a.name=name;

7. Care dintre următoarele definiții de tip sunt corecte sintactic?

```
a) type info = record
  a,b:real;
  ma,mg: byte;
end;
```

```
a) typedef struct {
  float a,b;
  int ma,mg;
} info;
```

```
b) type record = record
  a,b:real;
  ma,mg:byte;
end;
```

```
c) type numar = record
  a,b:real;
  medie:numar;
end;
```

```
d) type medie = record
  a,b:integer;
  medie:real;
end;
```

```
b) typedef struct {
  float a,b;
  int ma,mg;
} struct;
```

```
c) typedef struct {
  float a,b;
  numar medie;
} numar;
```

```
d) typedef struct {
  int a,b;
  float medie;
} medie;
```

8. Considerând următoarele definiții de tipuri, specificați care dintre instrucțiunile de mai jos pot fi folosite pentru citirea câmpurilor variabilei *exemp*?

```
Type
  pers = record
    name: string; age: integer
  end;
  ap = record
    locatar: pers;
    nr: integer;
  end;
Var exemp: ap;
```

```
typedef struct {
  char name[256];
  int age;
} pers;
typedef struct {
  pers locatar;
  int nr;
} ap;
ap exemp;
```

```
a) with exemp do begin
  readln(locatar.name);
  readln(locatar.age, nr)
end;
```

```
b) with exemp do begin
  with locatar do begin
    readln(name);
    read(age)
  end;
  read(nr)
end;
```

```
c) with exemp do begin
  readln(locatar.name);
  readln(locatar.age);
  readln(locatar.nr)
end;
```

```
d) with exemp do begin
  with locatar do begin
    readln(name); read(age)
  end;
  read(nr)
end;
```

```
a)
cin>>exemp.locatar.name;
cin>>exemp.locatar.age;
cin>>exemp.nr;
```

```
b)
pers *t=&exemp.locatar;
cin>>t->name;
cin>>t->age;
cin>>exemp.nr;
```

```
c)
cin>>exemp.locatar.name;
cin>>exemp.locatar.age;
cin>>exemp.locatar.nr;
```

```
d)
pers *t=&exemp.locatar;
cin>>t->name;
cin>>t->age;
cin>>t->nr;
```

9. Considerând următoarele definiții de tipuri, specificați care dintre atribuiri de mai jos sunt corecte sintactic?

```

Type
d_c = record
  zi, luna, an: integer;
end;
stamp = record
  data: d_c;
  time : record
    h, min, sec: integer;
  end;
end;

```

var a: stamp; b: d_c;

- a) a.time.h:=23;
- b) a.d_c.zi:=23;
- c) b.zi:=b.luna;
- d) b.data:=23;

```

typedef struct {
  int zi, luna, an;
} d_c;
typedef struct {
  d_c data;
  struct {
    int h, min, sec;
  } time;
} stamp;

stamp a; d_c b;

```

- a) a.time.h=23;
- b) a.d_c.zi=23;
- c) b.zi=b.luna;
- d) b.data=23;

10. Presupunem că următoarele declarații au fost făcute pentru a reține un polinom P de grad n . Specificați care dintre secvențele de mai jos calculează corect produsul dintre polinomul P și scalarul x :

```

Type
Monom=record
  cf: real; grad: integer;
end;
Polinom = array[1..35] of monom;
Var p: Polinom; x, i, n: integer;

```

- a) for i:=1 to n+1 do
p.cf:=p.cf * x;
- b) for i:=1 to n do
p[i].cf:=p[i].cf + x;
- c) for i:=1 to n+1 do
p[i].cf:=x.cf * p[i].cf;
- d) for i:=1 to n+1 do
p[i].cf:=p[i].cf*x;

```

typedef struct {
  float cf; int grad;
} Monom;
typedef Monom Polinom[35];
Polinom p;
int x, i, n;

```

- a) for (i=0; i<=n; i++)
p.cf*=x;
- b) for (i=0; i<n; i++)
p[i].cf+=x;
- c) for (i=0; i<=n; i++)
p[i].cf*=x.cf;
- d) for (i=0; i<=n; i++)
p[i].cf*=x;

11. Considerăm următoarele declarații:

```

type cerc=record
  o_x, o_y, raza: integer;
end;

```

```

typedef struct {
  int o_x, o_y, raza;
} cerc;

```

```

cilindru = record
  h, volum: integer;
  baza: cerc;
end;
var x: cilindru;
y: cerc;

```

```

typedef struct {
  int h, volum;
  cerc baza;
} cilindru;
cilindru x;
cerc y;

```

Care dintre următoarele atribuiri sunt corecte sintactic:

- a) x.y.raza := 10;
- b) x.h.raza := 10;
- c) x.baza.raza := 10;
- d) x.volum.o_x := 10;

- a) x.y.raza = 10;
- b) x.h.raza = 10;
- c) x.baza.raza = 10;
- d) x.volum.o_x = 10;

12. Considerăm următoarele declarații care permit memorarea în variabila a a unui polinom de grad n (<50), iar în variabila b a unui monom. Pentru fiecare din cele $n+1$ monoame ale polinomului se reține coeficientul și gradul său. Memorarea acestora nu este ordonată după grade.

```

type monom = record
  cf: integer;
  gr: integer; end;
type
  polinom=array[1..50] of monom;
var
  a: polinom;
  b: monom;
  i, n, m: integer;

```

```

typedef struct {
  float cf;
  int gr;
} monom;
typedef monom polinom[50];

polinom a;
monom b;
int i, n, m;

```

Identificați secvența de instrucțiuni care adună în mod corect monomul b la polinomul a :

- a) m:=1;
for i:=1 to n+1 do
if a.gr[i]=b.gr then begin
inc(a.cf[i], b.cf); m:=0
end;
if m=1 then begin
inc(n); a[n]:=b
end;
- b)
for i:=1 to n+1 do
inc(a[i].cf, b.cf);
- c) m:=1;
for i:=1 to n+1 do
if a[i].gr=b.gr then begin
inc(a[i].cf, b.cf); m:=0
end;
if m=1 then begin
inc(n); a[n]:=b
end;

- a) m:=1;
for (i=0; i<=n; i++)
if (a.gr[i]==b.gr) {
a.cf[i]+=b.cf; m:=0;
}
if (m) {
a[++n]=b;
}
- b)
for (i=0; i<=n; i++){
a[i].cf[i]+=b.cf;
}
- c) m:=1;
for (i=0; i<=n; i++)
if (a[i].gr==b.gr){
a[i].cf+=b.cf;
m:=0;
}
if (m) a[++n]=b;

```
d) for i:=1 to n do
    if a[i].gr=b.gr then
        inc(a[i].cf,b.cf);
```

```
d) for (i=1; i<=n; i++)
    if (a[i].gr==b.gr)
        a[i].cf+=b.cf;
```

13. Considerăm următoarele declarații care permit memorarea în variabila *a* a unui polinom de grad *n* (<50), iar în variabila *b* a unui monom. Pentru fiecare din cele *n*+1 monoame ale polinomului se reține coeficientul și gradul său.

```
type monom = record
    cf: real;
    gr: integer; end;
type
    polinom=array[1..50] of monom;
var
    a:polinom;
    b:monom;
    i,n,m:integer;
```

```
typedef struct {
    float cf;
    int gr;
} monom;
typedef monom polinom[50];

polinom a;
monom b;
int i,n,m;
```

Identificați secvența de instrucțiuni care înmulțește în mod corect polinomul *a* cu monomul *b*:

```
a) for i:=1 to n+1 do
    a[i].gr:=a[i].gr * b.gr;

b) for i:=1 to n+1 do begin
    a[i].cf:=a[i].cf * b.cf;
    a[i].gr:=a[i].gr + b.gr;
end;

c) for i:=1 to n+1 do begin
    a[i].cf:=a[i].cf + b.cf;
    a[i].gr:=a[i].gr + b.gr;
end;

d) for i:=1 to n+1 do begin
    a.cf[i]:=a.cf[i] * b.cf;
    a.gr[i]:=a.gr[i] + b.gr;
end;
```

```
a) for (i=0; i<=n; i++)
    a[i].gr *= b.gr;

b) for (i=0; i<=n; i++){
    a[i].cf *= b.cf;
    a[i].gr += b.gr;
}

c) for (i=0; i<=n; i++){
    a[i].cf += b.cf;
    a[i].gr += b.gr;
}

d) for (i=0; i<=n; i++){
    a.cf[i] += b.cf;
    a.gr[i] *= b.gr;
}
```

14. Variabila *a* este utilizată într-un program pentru a memora numele și prenumele unui elev. Identificați care dintre declarațiile următoare este *incorectă*:

```
a) var a:
    record nm, pr:string[15] end;
b) var a:char[30];
c) var
    a:array[1..2] of string[150];
d) var a:string;
```

```
a) struct{
    char *nm, char *pr;}a;
b) float a[15];
c) char a[200];
d) char a[2][15];
```

1.2.2 Probleme rezolvate

1. La o școală primară sunt înscriși câte *m* elevi în fiecare dintre cele *n* clase (numerate de la 1 la *n*). Pentru fiecare clasă se cunosc câți elevi navetiști există și care este numărul de băieți. Scrieți o secvență de program care calculează numărul de fete din școală, care este procentul navetiștilor înscriși și afișează lista claselor care au procentul de navetiști mai mare decât procentul pe școală.

Exemplu:

Pentru *n*=2 și *m*=4;
Clasa 1: 2 navetiști, 1 băiat
Clasa 2: 1 navetist, 3 băieți

se va afișa:
Numărul de fete: 4
Procentajul navetiștilor: 37.5%
Lista: clasa 1

Soluție

Pentru fiecare clasă se vor reține, cu ajutorul unei înregistrări, numărul de navetiști (*nv*) și procentul acestora (*pc*) față de numărul total de elevi ai clasei. Numărul de fete (*f*) din școală se poate determina în momentul citirii datelor referitoare la fiecare clasă.

```
1 type c = record
2     nv:byte; pc:real;
3     end;
4 var a:array[1..50] of c;
5 x:real;
6 n,tnv,i,f,m,b:integer;
7 begin
8     readln(n,m);
9     for i:=1 to n do begin
10         readln(a[i].nv);
11         readln(b);
12         f:=f+m-b;
13         tnv:=tnv+a[i].nv;
14         a[i].pc:=a[i].nv*100/m;
15     end;
16 x:=tnv*100/(m*n);
17 writeln(f,' ',x:0:2);
18 for i:=1 to n do
19     if a[i].pc>x then write(i)
20     end;
```

```
#include <stdio.h>
typedef struct {
    int nv; float pc;
} c;
c a[50];
float x;
int n,tnv, i,f,m,b;
void main() {
    scanf("%d %d",&n,&m);
    for(i=0;i<n;i++){
        scanf("%d %d",&a[i].nv,&b);
        f+=m-b;
        tnv+=a[i].nv;
        a[i].pc=a[i].nv*100.0/m;
    }
    x=tnv*100.0/(m*n);
    printf("%d %.2f\n",f,x);
    for(i=0;i<n;i++)
        if (a[i].pc>x)
            printf("%d",i+1);
}
```

2. În laboratorul de informatică al unei școli se află *n* calculatoare (numerate de la 1 la *n*). Pentru fiecare se cunosc tipul procesorului (486, Pentium I, II, III, IV, Duron, etc.), frecvența procesorului (exprimată în MHz), memoria RAM (exprimată în MB) și capacitatea hard-diskului (exprimată în MB).

Realizați un program care afișează calculatoarele care pot fi conectate într-o rețea sub un sistem de operare (frecvența de cel puțin *x* Mhz, memoria de cel puțin *y* MB și capacitatea hard-diskului de cel puțin *z* MB).

Ca server va fi ales un calculator (eventual procesor Pentium) cu cele mai bune performanțe în ordinea: Frecvență, memorie RAM.

Soluție

Datele referitoare la fiecare calculator vor fi preluate dintr-o înregistrare, ale cărei câmpuri vor reține tipul procesorului (*p*), frecvența procesorului (*fr*), memoria RAM (*ram*) și capacitatea hard-diskului (*hdd*). În procesul de citire a datelor se va identifica și serverul rețelei.

```

1  type c=record
2    fr,ram,hdd:integer;p:string;
3  end;
4  var a:array[1..50]of c;
5  i,n,x,y,z,s,mfr,mram:integer;
6  begin
7    readln(n,x,y,z);
8    mfr:=0; mram:=0;
9    for i:=1 to n do
10     with a[i] do begin
11       readln(p);
12       readln(fr,ram,hdd);
13       if (fr>mfr)or
14         (fr=mfr)and(ram>mram) then
15         begin
16           s:=i; mfr:=fr; mram:=ram;
17         end;
18     end;
19     writeln('serverul:',s);
20     for i:=1 to n do
21       with a[i] do begin
22         if (i<>s)and(fr>x)and
23           (ram>y)and(hdd>z)
24         then writeln('statia ',i);
25       end;
26     end.

```

```

#include <stdio.h>
typedef struct {
  int fr,ram,hdd; char p[256];
} c;
c a[50]; int i,n,x,y,z,s,mfr,
mram;
void main() {
  scanf("%d%d%d%d",&n,&x,&y,&z);
  mfr=mram=0;
  for (i=0;i<n;i++){
    scanf("%s%d%d%d",&a[i].p,
      &a[i].fr,&a[i].ram,&a[i].hdd);
    if (a[i].fr>mfr||
      (a[i].fr==mfr&&a[i].ram>mram)){
      s=i; mfr=a[i].fr;
      mram=a[i].ram;
    }
  }
  printf("serverul:%d",s+1);
  for(i=0;i<n;i++){
    if (i!=s&&a[i].fr>x&&
      a[i].ram>y&&a[i].hdd>z)
      printf("statia %d\n",i+1);
  }
}

```

3. Se consideră o listă formată din date referitoare la *n* elevi. Fiecărui elev *i* se cunoaște numele și prenumele, media notelor de la oral la disciplina Informatică și nota din teză. Realizați un program care afișează pentru fiecare literă a alfabetului, numele și prenumele elevului care a obținut cea mai mare medie dintre cei ai căror nume de familie începe cu acea literă.

Soluție

Pentru fiecare elev se vor reține într-o înregistrare numele (*np*) și media semestrială la informatică (*m*). Vectorul ce reține datele tuturor elevilor va fi ordonat descrescător în funcție de medie.

Pentru fiecare literă a alfabetului se va identifica prima înregistrare din vector pentru care primul caracter al câmpului *np* este egal cu litera curentă.

```

1  type e=record
2    np:string; m:real;
3  end;
4  var a:array[1..50]of e;
5  x:e; nt,j,n,i:integer;
6  mo:real; nm,pr:string;
7  s:array['A'..'Z']of boolean;
8  begin
9    readln(n);
10   for i:=1 to n do begin
11     readln(nm);
12     readln(pr);
13     a[i].np:=nm+' '+pr;
14     readln(mo,nt);
15     a[i].m:=(mo*3+nt)/4;
16   end;
17   for i:=1 to n-1 do
18     for j:=i+1 to n do
19       if a[i].m<a[j].m then begin
20         x:=a[i];a[i]:=a[j];a[j]:=x;
21       end;
22     i:=1;
23     while i<=n do begin
24       if not s[a[i].np[1]] then
25         begin
26           writeln(a[i].np);
27           s[a[i].np[1]]:=true;
28         end;
29       inc(i);end;
30   end.

```

```

#include <stdio.h>
#include <string.h>
typedef struct {
  char np[256]; float m;
} e;
e a[50];x;
int nt,i,j,n,s[256];
float mo;
char nm[256],pr[256];
void main() {
  scanf("%d",&n);
  for(i=0;i<n;i++){
    scanf("%s%s",nm,pr);
    strcat(a[i].np,nm);
    strcat(a[i].np," ");
    strcat(a[i].np,pr);
    scanf("%f%d",&mo,&nt);
    a[i].m:=(mo*3.0+nt)/4.0;
  }
  for(i=0;i<n;i++)
    for(j=i+1;j<n;j++){
      if (a[i].m<a[j].m){
        x=a[i];a[i]=a[j];a[j]=x;
      }
    }
  for(i=0;i<n;i++)
    if (!s[a[i].np[0]]){
      printf("%s\n",a[i].np);
      s[a[i].np[0]]=1;
    }
}

```

4. La un magazin au fost aduse *n* sortimente de produse (numerotate de la 1 la *n*), pentru fiecare cunoscându-se cantitatea (exprimată în bucăți), prețul de achiziție și adaosul comercial (exprimat în procent din valoarea de achiziție). În decursul unei săptămâni s-a contorizat, pe zile, cantitatea vândută din fiecare produs. Realizați un program care afișează valoarea maximă a vânzărilor obținute într-o zi și valoarea vânzărilor săptămânale ale unui produs *x*.

Soluție

O înregistrare care reține datele referitoare la un produs va memora următoarele: valoarea de vânzare a produsului (*v*), cantitatea achiziționată (*c*) și vânzările efectuate în fiecare zi a săptămânii (vectorul *z*). Pentru fiecare zi a săptămânii se va calcula valoarea produselor vândute, actualizându-se valoarea maximă a vânzărilor realizată într-o zi.

```

1  type p=record
2    v:real; c:byte;
3    z:array[1..7]of real;
4  end;
5  var a:array[1..50]of p;
6  j,n,i,x:integer;
7  t,max,s,pr,ad:real;

```

```

#include <stdio.h>
typedef struct {
  float v,z[7]; int c;
} p;
p a[50];
int i,j,n,x;
float t,max,s,pr,ad;

```

```

8 begin
9 readln(n,x); max:=0;
10 for i:=1 to n do begin
11 readln(a[i].c);
12 readln(pr); readln(ad);
13 a[i].v:=pr+pr*ad/100;
14 for j:=1 to 7 do
15 read(a[i].z[j])
16 end;
17 for i:=1 to 7 do begin
18 t:=0;
19 for j:=1 to n do
20 t:=t+a[j].v*a[j].z[i];
21 if t>max then max:=t;
22 end;
23 for i:=1 to 7 do
24 s:=s+a[x].z[i]*a[x].v;
25 writeln(max:0:2, ' ', s:0:2);
26 end.

```

```

void main(){
scanf("%d%d",&n,&x); max=0.0;
for(i=0;i<n;i++){
scanf("%d",&a[i].c);
scanf("%f%f",&pr,&ad);
a[i].v=pr+pr*ad/100.0;
for (j=0;j<7;j++){
scanf("%f",&a[i].z[j]);
}
for(i=0;i<7;i++){
for(t=j=0;j<n;j++){
t+=a[j].v*a[j].z[i];
if (t>max) max=t;
}
x--;
for(i=0;i<7;i++){
s+=a[x].z[i]*a[x].v;
}
printf("%.2f %.2f\n",max,s);
}
}

```

5. Realizați un program pentru evidența studenților unei facultăți, care să permită alegerea repetată a uneia dintre opțiunile de mai jos, păstrându-se lista studenților ordonată după nume:

- adăugarea unui student în grupă;
- afișarea informațiilor despre un anumit student, căutat dupe nume;
- listarea tuturor studenților din grupă.

Despre fiecare student al unei grupe se cunosc numele studentului, media la sfârșitul unei sesiuni și valoarea bursei.

Soluție

Lista cu datele referitoare la studenți va fi reținută într-un vector de înregistrări (a), care pe tot parcursul procesului de prelucrare va fi ordonat crescător după câmpul nume (np).

Operația de adăugare a unui nou student se va efectua în două etape: se identifică printr-o parcurgere poziția corectă unde trebuie plasată noua înregistrare, după care se va efectua o operație de deplasare spre dreapta a elementelor următoare.

Operația de căutare a unei înregistrări este realizată prin algoritmul de căutare binară, deoarece lista este ordonată.

```

1 type s=record
2 m,b:integer;np:string; end;
3 var a:array[1..50]of s;
4 x,i,m,j,n:integer;
5 nm,pr:string;
6 ok:boolean;
7 begin
8 write('add=1; search=2;');
9 writeln('list=3; exit=4. ');
10 n:=0;

```

```

#include <stdio.h>
#include <string.h>
typedef struct {
int m,b; char np[256];
} s;
s a[50]; int x,i,m,j,n,ok;
char nm[256],pr[256];
void main() {
printf("add=1; search=2;");
printf("list=3; exit=4.\n");
}

```

```

11 repeat
12 writeln('Operatia=?');
13 readln(x);
14 case x of
15 1: begin readln(nm);
16 readln(pr);
17 i:=0;
18 repeat
19 inc(i);
20 until (i>n) or (a[i].np>nm+pr);
21 for j:=n downto i do
22 a[j+1]:=a[j];
23 readln(a[i].m,a[i].b);
24 a[i].np:=nm+pr;
25 inc(n);
26 end;
27 2: begin
28 readln(nm); ok:=true;
29 i:=1; j:=n;
30 while (i<=j) and ok do begin
31 m:=(i+j)div 2;
32 if a[m].np=nm then begin
33 writeln(a[m].m,a[m].b);
34 ok:=false;
35 end
36 else
37 if a[m].np>nm then j:=m-1
38 else i:=m+1
39 end
40 end;
41 3: for i:=1 to n do
42 writeln(a[i].np);
43 end;
44 until x=4;
45 end.

```

```

do {
printf("Operatia=?");
scanf("%d",&x);
switch (x){
case 1:
scanf("%s%s",nm,pr);
strcat(nm,pr);
for(i=0;i<n&&
strcmp(a[i].np,nm)<=0;i++){
for(j=n-1;j>=i;j--)
a[j+1]=a[j];
scanf("%d%d",&a[i].m,&a[i].b);
strcpy(a[i].np,nm);n++;
break;
case 2:
scanf("%s",&nm);ok=1;
for(i=0,j=n-1;i<=j&&ok){
m=(i+j)/2;
if (!strcmp(a[m].np,nm)){
printf("%d %d\n",
a[m].m,a[m].b);ok=0;
} else
if (strcmp(a[m].np,nm)>0)
j=m-1;
else i=m+1;
}
break;
case 3:
for(i=0;i<n;i++)
printf("%s\n",a[i].np);
break;
} while (x!=4);
}

```

6. Pentru elevii clasei a XII-a, profesorul diriginte are nevoie de următoarele informații, pentru a calcula nota la purtare a unui elev: numele, prenumele, numărul total de absențe și numărul de absențe motivate. Pentru fiecare 10 absențe nemotivate, elevul pierde un punct la nota de la purtare. Dacă numărul absențelor nemotivate este mai mare de 50, elevul primește media 4. Realizați un program care calculează notele la purtare ale elevilor și afișează o listă a elevilor ordonată descrescător după media de la purtare.

Exemplu n=6

și lista (nume/total absențe – motivate)

Ionescu Vlad	17 - 10
Popescu Ion	24 - 1
Ionescu Ana	8 - 6
Alexe Maria	20 - 4
Mitea Ilie	73 - 6
Popescu Dan	12 - 12

se va afișa:

Ionescu Ana	10
Ionescu Vlad	10
Popescu Dan	10
Alexe Maria	9
Popescu Ion	8
Mitea Ilie	4

Soluție

O înregistrare ce memorează datele referitoare la un elev are două câmpuri în care se rețin numele și prenumele (*np*) și media la purtare (*m*). Vectorul *a* în care se rețin datele despre elevii clasei va fi ordonat descrescător în funcție de câmpul medie.

```
1 type e=record
2   m:integer;np:string;
3   end;
4 var a:array[1..50]of e;
5   i,j,n,nta,nam:integer;
6   nm,pr:string; x:e;
7
8 begin
9   readln(n);
10  for i:=1 to n do begin
11    readln(nm);
12    readln(pr);
13    a[i].np:=nm+' '+pr;
14    readln(nta,nam);
15    a[i].m:=10-(nta-nam)div 10;
16    if a[i].m<5 then a[i].m:=4;
17  end;
18  for i:=1 to n-1 do
19    for j:=i+1 to n do
20      if a[i].m<a[j].m then
21        begin
22          x:=a[i];a[i]:=a[j];
23          a[j]:=x;
24        end;
25    for i:=1 to n do
26      writeln(a[i].np,' ',a[i].m);
27 end.
```

```
#include <stdio.h>
#include <string.h>
typedef struct {
  int m; char np[256];
} e;
e a[50]; int i,j,n,nta,nam;
char nm[256],pr[256];
void main(){
  scanf("%d",&n);
  for(i=0;i<n;i++){
    scanf("%s%s",nm,pr);
    strcat(a[i].np,nm);
    strcat(a[i].np," ");
    strcat(a[i].np,pr);
    scanf("%d%d",&nta,&nam);
    a[i].m=10-(nta-nam)/10;
    if (a[i].m<5) a[i].m=4;
  }
  for(i=0;i<n;i++){
    for(j=i+1;j<n;j++){
      if (a[i].m<a[j].m){
        x=a[i];a[i]=a[j];a[j]=x;
      }
    }
  }
  for(i=0;i<n;i++){
    printf("%s %d\n",
      a[i].np,a[i].m);
  }
}
```

7. Se consideră n intervale închise $[a,b]$, a și b numere întregi. Să se determine reuniunea acestora.

Exemplu $n=5$ și intervalele:

2 4	1 4
1 3	5 9
5 8	10 12
10 12	
6 9	

Soluție

Vom reține intervalele într-un tablou unidimensional de înregistrări. Primul pas al algoritmului va ordona intervalele în funcție de capătul stâng al acestora. Pentru determinarea reuniunii, este suficientă o parcurgere liniară a acestora, păstrând la fiecare moment capătul stâng și drept al intervalului curent al reuniunii, actualizând la nevoie valorile acestora.

```
1 type it=record
2   x,y:integer; end;
3 var a:array[1..100]of it;
4 rx,ry,i,n,j:integer; t:it;
5 begin
6   readln(n);
7   for i:=1 to n do
8     read(a[i].x,a[i].y);
9   for i:=1 to n-1 do
10    for j:=i+1 to n do
11      if a[i].x>a[j].x then begin
12        t:=a[i];
13        a[i]:=a[j];
14        a[j]:=t;
15      end;
16    rx:=a[1].x; ry:=a[1].y;
17    for i:=2 to n do
18      if a[i].x>ry then begin
19        writeln(rx,' ',ry);
20        rx:=a[i].x;ry:=a[i].y;
21      end
22    else
23      if a[i].y>ry then
24        ry:=a[i].y;
25    writeln(rx,' ',ry)
26 end.
```

```
#include <stdio.h>
typedef struct {
  int x,y;
} it;
it a[100]; int rx,ry,i,j,n;
void main() {
  scanf("%d",&n);
  for(i=0;i<n;i++){
    scanf("%d%d",&a[i].x,&a[i].y);
  }
  for(i=0;i<n;i++){
    for(j=i+1;j<n;j++){
      if (a[i].x>a[j].x) {
        t=a[i];
        a[i]=a[j];
        a[j]=t;
      }
    }
    rx=a[0].x; ry=a[0].y;
    for (i=1;i<n;i++){
      if (a[i].x>ry)
      {
        printf("%d %d\n",rx,ry);
        rx=a[i].x;ry=a[i].y;
      } else
      if (a[i].y>ry) ry=a[i].y;
    }
    printf("%d %d\n",rx,ry);
  }
}
```

8. Pentru fiecare dintre cei n elevi ai unei clase se cunoaște numărul matricol (număr format din maximum 8 cifre), anul, luna și ziua nașterii. Să se afișeze lista elevilor care își vor sărbătorii ziua de naștere anul acesta, cunoscându-se ziua și luna în care ne aflăm. Se vor afișa numărul matricol al elevului, ziua și luna nașterii ale acestuia. Elevii vor fi afișați în ordine crescătoare a datei nașterii. Datele de intrare se citesc din fișierul *date.in* în care pe prima linie se află trei numere n , ziua curentă și luna curentă. Pe următoarele n linii se află câte patru numere, în ordine reprezentând: numărul matricol, ziua, luna și anul nașterii fiecărui elev.

Exemplu: Pentru *date.in* se va afișa:

4 12 3	12301 21 4
23128 4 12 1969	23128 4 12
21398 23 1 1970	
12301 21 4 1970	
54312 11 3 1970	

Soluție

Pentru fiecare elev se va reține pe lângă numărul matricol (m), ziua (z), luna (l) și numărul zilei din an (t) în care își va sărbătorii ziua de naștere. Anul nașterii nu va fi preluat într-un câmp al înregistrării.

Vectorul a în care sunt reținute ca elemente aceste înregistrări, va fi ordonat crescător după câmpul t . După această operație, printr-o parcurgere a vectorului se va identifica prima înregistrare pentru care ziua curentă din an este mai mică decât valoarea din câmpul t .

```

1 type p=record
2   m,z,l,t :longint ;
3 end;
4 var a:array[1..50]of p;
5   s:p;i,n,j,x,y,t,d:integer;
6 begin
7   assign(input,'date.in');
8   reset(input); readln(n,x,y);
9   d:=x+(y-1)*30;
10  for i:=1 to n do
11    with a[i] do begin
12      readln(m,z,l);
13      t :=z+(l-1)*30 ;
14    end;
15  for i:=1 to n-1 do
16    for j:=i+1 to n do
17      if a[i].t>a[j].t then begin
18        s:=a[i];a[i]:=a[j]; a[j]:=s;
19      end;
20  i:=1;
21  while (a[i].t<d)and(i<=n)do
22    inc(i);
23  if i>n then
24    with a[1] do
25      writeln(m,' ',z,' ',l)
26  else for j:=i to n do
27    with a[j] do
28      writeln(m,' ',z,' ',l);
29 end.

```

```

#include <stdio.h>
typedef struct {
  long m,z,l,t;
} p;
p a[50],s;
int i,n,j,x,y,t,d;
void main() {
  freopen("date.in","r",stdin);
  scanf("%d%d%d",&n,&x,&y);
  d=x+(y-1)*30;
  for(i=0;i<n;i++){
    scanf("%ld%ld%ld%ld",&a[i].m,
      &a[i].z,&a[i].l,&a[i].t);
    a[i].t=a[i].z+(a[i].l-1)*30;
  }
  for (i=0;i<n;i++)
    for (j=i+1;j<n;j++)
      if (a[i].t>a[j].t) {
        s=a[i]; a[i]=a[j]; a[j]=s;
      }
  for(i=0;i<n&& a[i].t<d; i++);
  if (i>n)
    printf("%ld %ld %ld\n",
      a[0].m,a[0].z,a[0].l);
  else
    for(j=i;j<n;j++)
      printf("%ld %ld %ld\n",
        a[j].m,a[j].z,a[j].l);
}

```

9. Într-o sală a unui teatru sunt programate mai multe spectacole. Pentru fiecare dintre ele se cunoaște intervalul orar al desfășurării (ora de început și cea de sfârșit). Deoarece unele dintre ele se suprapun ca ore de desfășurare, se dorește eliminarea unui număr minim de spectacole astfel încât cele rămase să se poată susține într-o singură zi, în sala de teatru avută la dispoziție.

Datele de intrare se citesc din fișierul *teatru.in*, unde pe prima linie se găsește numărul n , iar pe următoarele n linii se găsesc două numere reale exprimate cu două zecimale (*hh.mm*) reprezentând ora și minutele momentului de început și de sfârșit al fiecărui spectacol. Se vor afișa intervalele orare în care erau programate să se desfășoare spectacolele care au fost eliminate.

Exemplu: Pentru *teatru.in*

5	se va afișa:
10.00 12.30	9.30 13.20
9.30 13.20	12.20 15.30
15.00 16.00	
12.20 15.30	
12.30 14.20	

Soluție

Pentru fiecare spectacol se va reține într-o înregistrare momentul de început (x) și cel de sfârșit (y).

Tabloul a ce memorează cele n înregistrări va fi ordonat crescător după x iar pentru spectacole cu același moment de început sortarea se va face crescător după y . Într-o parcurgere liniară se vor afișa spectacolele eliminate.

```

1 type s=record x,y:real; end;
2 var a:array[1..50] of s;
3 t:s; u,i,n,j:integer; sf:real;
4 begin
5   assign(input,'teatru.in');
6   reset(input); readln(n);
7   for i:=1 to n do
8     readln(a[i].x,a[i].y);
9   for i:=1 to n-1 do
10    for j:=i+1 to n do
11      if (a[i].x > a[j].x) or
12      (a[i].x=a[j].x)and(a[i].y>a[j].y)
13      then begin
14        t:=a[i];
15        a[i]:=a[j];
16        a[j]:=t;
17      end;
18  u:=1; sf:=a[1].y;
19  for i:=2 to n do
20    if a[i].y<sf then
21      begin
22        writeln(a[u].x:0:2,a[u].y:0:2);
23        u:=i;
24        sf:=a[i].y;
25      end
26    else
27      if a[i].x<sf then
28        writeln(a[i].x:0:2,a[i].y:0:2)
29      else begin
30        u:=i;
31        sf:=a[i].y;
32      end
33  end.

```

```

#include <stdio.h>
typedef struct {
  float x,y;
} s;
s a[50],t;int i,j,n,u;float sf;
void main(){
  freopen("teatru.in","r",stdin);
  scanf("%d",&n);
  scanf("%f%f",&a[0].x,&a[0].y);
  for (i=1;i<n;i++)
    scanf("%f%f",&a[i].x,&a[i].y);
  for(i=0;i<n;i++)
    for(j=i+1;j<n;j++)
      if(a[i].x>a[j].x||(a[i].x==
        a[j].x&& a[i].y>a[j].y)){
        t=a[i];
        a[i]=a[j];
        a[j]=t;
      }
  u=0;sf=a[0].y;
  for(i=1;i<n;i++)
    if (a[i].y<sf){
      printf("%.2f %.2f\n",
        a[u].x,a[u].y);u=i;sf=a[i].y;
    } else
      if (a[i].x<sf)
        printf("%.2f %.2f\n",
          a[i].x,a[i].y);
    else {
      u=i;
      sf=a[i].y;
    }
}

```

10. Se consideră un șir de n intervale de forma $[a_i, b_i]$, cu a_i, b_i numere întregi. Un interval poate fi eliminat din șirul celor n dacă există un alt interval care îl include pe acesta. Determinați numărul maxim de intervale care pot fi eliminate. În fișierul text *interval.in* se găsește pe prima linie numărul n ($n < 16000$), iar pe următoarele n linii, perechi de numere naturale, mai mici decât 2000000000 ce reprezintă capetele intervalelor. Rezultatul va fi afișat pe ecran.

Exemplu: *interval.in*

5	Se va afișa
0 10	3
2 9	
3 8	
1 15	
6 11	

Soluție

Se sortează crescător șirul intervalelor după limita inferioară și descrescător după limita superioară. La parcurgerea intervalelor se va actualiza cel mai mare capăt din dreapta (*maxdr*). Orice interval pentru care capătul din dreapta este mai mic decât *maxdr* este redundant, deci inclus în altul.

```
1  type s = record
2      x, y: longint;
3  end;
4  var a: array[1..50] of s;
5  t: s; nr, i, n, j, md: integer;
6  begin
7      assign(input, 'interval.in');
8      reset(input); readln(n);
9      for i := 1 to n do
10         with a[i] do readln(x, y);
11     for i := 1 to n - 1 do
12         for j := i + 1 to n do
13             if (a[i].x > a[j].x) or
14                 (a[i].x = a[j].x and (a[i].y < a[j].y))
15             then begin
16                 t := a[i]; a[i] := a[j]; a[j] := t;
17             end;
18     nr := 0;
19     md := a[1].y;
20     for i := 2 to n do
21         if a[i].y < md then inc(nr);
22         else md := a[i].y;
23     writeln(nr);
24 end.
```

```
#include <stdio.h>
typedef struct {
    long x, y;
} s;
s a[50], t; int nr, i, n, j, md;
void main() {
    freopen("interval.in", "r",
        stdin); scanf("%d", &n);
    for(i=0; i<n; i++)
        scanf("%ld%ld",
            &a[i].x, &a[i].y);
    for(i=0; i<n; i++)
        for(j=i+1; j<n; j++)
            if(a[i].x > a[j].x || (a[i].x ==
                a[j].x && a[i].y < a[j].y)){
                t = a[i]; a[i] = a[j]; a[j] = t;
            }
    nr = 0;
    md = a[0].y;
    for(i=1; i<n; i++)
        if (a[i].y < md) nr++;
        else md = a[i].y;
    printf("%d\n", nr);
}
```

1.2.3 Probleme propuse

1. Se consideră o listă formată din datele referitoare la n elevi. Fiecărui elev i se cunoaște numele și prenumele, media notelor de la oral la disciplina Informatică și nota din teză. Să se creeze un program care afișează media pe clasă la disciplina informatică și numele elevilor care au obținut cea mai mare medie semestrială.
2. Se consideră o listă formată din datele referitoare la n elevi. Fiecărui elev i se cunoaște numele și prenumele, media notelor de la oral la disciplina Informatică și nota din teză. Creați un program care afișează în ordinea descrescătoare a mediilor, numele și prenumele elevilor care au promovat la această disciplină.
3. Se consideră o listă formată din datele referitoare la n elevi. Fiecărui elev i se cunoaște numele și prenumele, media notelor de la oral la disciplina Informatică și nota din teză. Realizați un program care afișează în ordine alfabetică elevii corigenți la Informatică.

4. Pentru a scrie catalogul, dirigintele are nevoie de numele și prenumele elevilor. Să se ordoneze aceste date alfabetic, după nume, iar pentru elevii cu același nume să se ordoneze alfabetic după prenume.

Exemplu: $n = 6$ și elevii:

Ionescu Vlad, Popescu Ion,
Ionescu Ana, Alexe Maria,
Mitea Ilie, Popescu Dan.

se va afișa:

Alexe Maria, Ionescu Ana,
Ionescu Vlad, Mitea Ilie,
Popescu Dan, Popescu Ion

5. Se cunoaște numărul de sportivi participanți la o competiție oarecare. Pentru fiecare dintre ei se cunoaște data nașterii (luna și anul). Cunoșcându-se data (luna și anul) la care se desfășoară competiția, să se afișeze media de vârstă a sportivilor, exprimată în același mod. Afișați și lista datelor de naștere ale sportivilor cu vârsta mai mică decât cea medie.

6. Se citesc de la tastatură datele referitoare la m elevi: nume, prenume, numărul de membri ai familiei și venitul net lunar al familiei. Un elev primește bursă dacă venitul pe fiecare membru al familiei nu depășește o valoare limită L . Realizați un program care afișează elevii ce nu au dreptul la bursă, în ordine alfabetică.

7. Se consideră un șir de n fracții (numitor, numărător). Afișați numărul de fracții echivalente cu ultima citită.

Exemplu: Pentru $n=4$ și fracțiile „(3, 5), (36, 60), (2, 4), (12, 20)” se va afișa: „2” (prima și a doua fracție).

8. Pentru n numere complexe, cărora li se cunosc partea reală și cea imaginară, să se afișeze în ordine crescătoare valorile modulelor ce nu aparțin intervalului $[a, b]$. Valorile reale a și b se citesc de la tastatură. Modulele rezultate vor fi afișate cu 3 zecimale.

9. Pentru o clasă de n elevi se cunosc următoarele date: numele, prenumele, vârsta și înălțimea. Se dorește realizarea unui tabel care să cuprindă doar elevii care au împlinit 14 ani și care au înălțimea cuprinsă în intervalul $[h_1, h_2]$. Elevii vor fi ordonați crescător după nume, iar în situația unor nume identice, vor fi ordonați crescător după prenume. Se va afișa pentru fiecare elev, numele, prenumele și înălțimea.

10. Se consideră un șir de n puncte în plan, pentru fiecare cunoscându-se coordonatele întregi (x, y) . Să se realizeze un program care identifică un pătrat de latură n , în interiorul căruia se află numărul maxim de puncte. Punctele aflate pe laturile pătratului se vor considera în „interiorul” lui. Se vor afișa coordonatele colțului stînga-jos al pătratului determinat, care va fi obligatoriu unul dintre cele n puncte date.

Exemplu: Pentru $n=5$ și punctele $(-10, -5), (4, 1), (3, 3), (1, 2), (10, 8)$ se va afișa $(1, 2)$.

11. Se consideră un șir de n puncte în plan, pentru fiecare cunoscându-se coordonatele întregi, (x, y) . Să se realizeze un program care determină numărul maxim de puncte coliniare situate pe o dreaptă paralelă cu axa OX .

Exemplu: Pentru $n=6$ și punctele „(-10,-5), (4,-5), (3,3), (1,2), (10,3), (5, 3)” se va afișa „3”.

12. Se consideră un șir de n puncte în plan, pentru fiecare cunoscându-se coordonatele întregi (x, y) . Să se realizeze un program care determină numărul de puncte care se află în afara unui triunghi, pentru care se cunosc coordonatele vârfurilor sale. Punctele aflate pe laturile triunghiului se vor considera în „interiorul” lui.

Exemplu: Pentru $n=6$, punctele „(0,1), (8,1), (4,2), (5,3), (5,8), (10, 3)” și triunghiul ale cărui vârfuri au coordonatele „(1,1), (5,10), (10,1)” se va afișa „2” (punctele (0,1) și (10, 3)).

13. Se consideră un șir de n fracții identificate prin numitor și numărător. Să se șteargă din șir toate fracțiile ireductibile. Fracțiile reductibile vor fi afișate în ordinea crescătoare a valorilor. În fișierul *in.txt* se va citi de pe fiecare linie perechea: numărător, numitor. Fracțiile rezultate vor fi afișate în același format în fișierul text *out.txt*.

Exemplu: Pentru fișierul *in.txt*

5
10 8
7 8
5 9
10 10
4 16

out.txt
4/16
10/10
10/8

14. Se consideră două fișiere *note.txt* și *nume.txt*. Unul conține pe fiecare linie câte două numere reprezentând notele la chimie ale unor elevi, iar pe liniile corespunzătoare din celălalt fișier se află numele acestora. Să se creeze un nou fișier, *final.txt*, în care pe fiecare linie să se regăsească numele elevului și media la chimie exprimată cu două zecimale. Cele două valori vor fi despărțite în cadrul liniilor prin câte un spațiu. Elevii vor fi scriși în ordinea descrescătoare a mediilor, iar la medii egale, crescător după nume.

Exemplu: *note.txt*

10 8
7 8
5 9
10 10
6 8
5 6

nume.txt

Ionescu M.
Mincu A.
Voicu B.
Micu M.
Udrea V.
Dan I.

final.txt

Micu M. 10.00
Ionescu M. 9.00
Mincu A. 7.50
Udrea V. 7.00
Voicu B. 7.00
Dan I. 5.50

15. Într-o școală există n clase de a XII-a, fiecare cu câte m elevi. Pentru fiecare dintre aceștia se cunosc: numele și prenumele, clasa din care face parte (identificată printr-o majusculă) și mediile semestriale.

Realizați un program care afișează:

- numele și prenumele șefului de promoție;
- lista pe clase a elevilor care nu vor susține examenul de bacalaureat, ordonați după nume și prenume.

În fișierul *date.txt* se vor citi, de pe prima linie, valorile lui n și m , apoi, de pe fiecare linie, informațiile referitoare la fiecare dintre cei $n*m$ elevi, în ordinea: clasa, nume, prenume, media semestru 1 și media semestru 2. Valorile vor fi despărțite prin câte un spațiu. Un elev care are o situație neîncheiată sau este corigent la o disciplină va avea media semestrială 0.00.

Afișarea rezultatelor se va face la ieșirea standard.

Exemplu: Pentru fișierul *date.txt*

2 2
A Ion Ion 7.00 9.00
A Mia Vlad 5.00 0.00
B Ilie Ana 0.00 10.00
B Savu Ion 10.00 9.00

se va afișa:

a) Șef de promoție: Savu Ion
b)
XII A: Mia Vlad
XII B: Ilie Ana

16. Se consideră un cinematograful în care există n săli (numerotate de la 1 la n), destinate vizionării filmelor. Se știe că fiecare sală se deschide publicului o singură dată pe zi, pentru o singură proiecție. Cunoscându-se cele n intervale orare $[a,b]$, (a și b numere reale) în care au loc proiecțiile în fiecare sală, să se identifice intervalul maxim în care toate sălile sunt deschise simultan publicului și lista sălilor în ordinea crescătoare a orei de început a proiecției.

Din fișierul text *film.in* se citesc datele de intrare în formatul următor: pe prima linie, numărul n de săli, iar pe următoarele n linii, perechi de numere reale cu două zecimale reprezentând ora și minutul începutului, respectiv sfârșitului proiecției filmului, în ordine începând cu sala 1.

Exemplu: Pentru fișierul *film.in*

4
13.30 15.00
12.45 16.45
11.30 14.30
13.00 14.00

se va afișa:

Toate filmele rulează simultan între:
13.30 14.00
Lista sălilor:
3, 2, 4, 1

1.3. Probleme de concurs ce procesează date structurare

1.3.1 Probleme rezolvate

1. (Bețe - **) Vacanța de primăvară îi oferă elevului Ionuț prilejul de a se odihni, de a citi și de a se juca. Este prea scurt timpul pentru a-și mai face și teme... Printre „jocurile clasice” ale părinților, a descoperit și un joc cu bețeșoare. Tabla de joc are de-a lungul ei, pe mijloc, un șanț cu poziții numerotate de la 1 la $L \leq 32767$. Pe tablă sunt plasate, în șanț, bețeșoare cu capătul stâng într-o anumită poziție. Bețele au lungimi diferite. Regula jocului este de a elimina cât mai puține bețe pentru a obține un număr maxim de bețe care nu se ating. Scrieți un program care să determine numărul maxim de bețe pe care Ionuț le poate obține.

Fișierul de intrare *bete.in* conține pe prima linie o valoare întreagă $n < 5000$, reprezentând numărul de bețe așezate pe tabla de joc, iar pe următoarele n linii câte două valori *poz* și *lung* ($poz + lung < 32767$), separate printr-un singur spațiu, reprezentând poziția pe tablă și respectiv lungimea fiecărui băț.

Fișierul de ieșire *bete.out* va conține o singură valoare reprezentând numărul maxim de bețișoare rămase pe tablă astfel încât acestea să nu se atingă.

Exemplu:

bete.in		bete.out
4	2	
1 1		
2 1		
3 1		
4 1		

Soluție:

Fiecare băț așezat pe tabla de joc poate reprezenta un interval de poziții $[poz_i, poz_i + lung_i]$ pe care le ocupă pe tablă. Astfel, problema se reduce la a determina o submulțime de intervale care nu se intersectează. Se sortează intervalele după capătul drept și se adaugă intervalele în soluție după o strategie *greedy*.

```
1 type interval=record poz,lung:integer; end;
2 var n,i,j,nr:integer; t:interval;
3   a:array[1..5000] of interval;
4 begin
5   assign(input,'bete.in'); reset(input); read(n);
6   for i:=1 to n do begin
7     read(a[i].poz,a[i].lung);
8     a[i].lung:=a[i].lung+a[i].poz;
9   end;
10  for i:=1 to n do
11    for j:=i+1 to n do
12      if a[i].lung>a[j].lung then begin
13        t:=a[i]; a[i]:=a[j]; a[j]:=t; end;
14  nr:=1; j:=1;
15  for i:=2 to n do
16    if a[j].lung<a[i].poz then begin inc(nr); j:=i; end;
17  assign(output,'bete.out'); rewrite(output); writeln(nr);
18 end.
```

```
1 #include <stdio.h>
2 typedef struct { int poz,lung; } interval;
3 interval a[5000]; int n;
4 void main() {
5   int i,j,nr;
6   freopen("bete.in","r",stdin); scanf("%d",&n);
7   for (i=0;i<n;i++) {
8     scanf("%d%d",&a[i].poz,&a[i].lung); a[i].lung+=a[i].poz;
9   }
10  for (i=0;i<n;i++)
11    for (j=i+1;j<n;j++)
12      if (a[i].lung>a[j].lung)
13        (t=a[i]; a[i]=a[j]; a[j]=t);
14  for (nr=1,i=1,j=0;i<n;i++)
15    if (a[j].lung<a[i].poz) { nr++; j=i; }
16  freopen("bete.out","w",stdout); printf("%d\n",nr);
17 }
```

2. (*Camion - *****) La firma la care lucrează Gigel există $M \leq 600$ tipuri de camioane, din fiecare tip existând $N \leq 600$ exemplare. Gigel așează camioanele firmei pe N rânduri, așezând pe fiecare coloană numai camioane de același tip. Se formează astfel o matrice în care liniile sunt numerotate de sus în jos de la 1 la N , iar coloanele sunt numerotate de la stânga la dreapta de la 1 la M . În fiecare noapte vine o bandă de hoți. Șeful bandei anunță: "în noaptea aceasta vom fura toate camioanele care se află în zona dreptunghiulară având colțul stânga-sus pe linia $x1$ și coloana $y1$, iar colțul opus pe linia $x2$ și coloana $y2$. În dimineața următoare, Gigel vede acest lucru, și „acoperă” furtul: pe fiecare linie în care există spații libere, deplasează spre stânga toate camioanele care se află în dreapta locului liber rămas. De exemplu, pentru $N=3$ și $M=5$ inițial avem următoarea amplasare:

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

În prima noapte hoții fură camioane, din dreptunghiul cu colțul stânga-sus în linia 2, coloana 2 și colțul dreapta-jos linia 3 coloana 3. Astfel, în ziua următoare, după ce Gigel deplasează camioanele, amplasarea va fi următoarea:

1	2	3	4	5
1	4	5		
1	4	5		

Cunoscând câte tipuri de camioane există inițial la firmă, pe câte rânduri au fost așezate, numărul $K \leq 30000$ de zile în care au loc furturi și coordonatele dreptunghiurilor din care fură hoții în fiecare noapte, determinați ce tipuri de camioane se află pe o anumită coloană din amplasarea finală.

Fișierul de intrare *camion.in* conține pe prima linie 4 numere naturale: $N M K$ și C , reprezentând numărul de rânduri pe care au fost așezate camioanele, numărul de coloane, numărul de nopți în care vor fura hoții camioane, respectiv numărul coloanei pentru care se dorește să se afle ce tipuri de camioane conține la final. Pe fiecare dintre următoarele K linii se vor afla câte 4 numere naturale. Pe linia $i+1$ se află $x1 y1 x2 y2$, $(x1,y1)$, reprezentând linia și coloana colțului stânga-sus, iar $(x2,y2)$ linia și coloana colțului dreapta-jos al dreptunghiului din care fură hoții în noaptea i . Nu este obligatoriu ca dreptunghiul din care se va efectua un furt să conțină camioane în fiecare loc. Numerele situate pe aceeași linie sunt separate prin câte un spațiu.

Fișierul de ieșire, *camion.out*, va conține N linii, pe fiecare câte un număr întreg. Numărul de pe linia i va reprezenta tipul camionului de pe linia i și coloana C , după K zile. În caz că pe linia i nu se găsește nici un camion, se va afișa valoarea 0 pe linia respectivă.

Exemplu:

camion.in		camion.out
3 5 3 1	3	
2 2 3 3	5	
1 1 3 2	5	
1 2 3 4		

Soluție: Se rezolvă problema separat pentru fiecare linie, astfel transformându-se problema într-una unidimensională. Pentru a rezolva problema unidimensională, se procesează operațiile în ordine inversă, transformând procesul de eliminare într-unul de inserare. Cum se cere doar coloana C, la fiecare pas se ține cont doar de elementul care se va afla pe acea coloană. Dimensiunile datelor impun folosirea compilatoarelor Free Pascal sau GCC.

```

1 type op=record x1,y1,x2,y2:integer; end;
2 var n,m,k,c,i,j,t:integer; a:array[1..30000] of op;
3 begin
4   assign(input,'camion.in'); reset(input);
5   read(n,m,k,c);
6   for i:=1 to k do
7     read(a[i].x1,a[i].y1,a[i].x2,a[i].y2);
8   assign(output,'camion.out'); rewrite(output);
9   for i:=1 to n do begin
10    t:=c;
11    for j:=k downto 1 do
12      if (a[j].x1<=i)and(i<=a[j].x2)and(a[j].y1<=t) then
13        t:=t+a[j].y2-a[j].y1+1;
14      if t<=m then writeln(t) else writeln(0);
15    end;
16  end.

```

```

1 #include <stdio.h>
2 typedef struct { int x1,y1,x2,y2; } op;
3 op a[30000]; int n,m,k,c;
4 void main() {
5   int i,j,t;
6   freopen("camion.in", "r", stdin);
7   scanf("%d%d%d%d",&n,&m,&c,&k);
8   for (i=0;i<k;i++)
9     scanf("%d%d%d%d",&a[i].x1,&a[i].y1,&a[i].x2,&a[i].y2);
10  freopen("camion.out","w",stdout);
11  for (i=1;i<=n;i++) {
12    for (t=c, j=k-1;j>=0;j--)
13      if (a[j].x1<=i&&i<=a[j].x2&&a[j].y1<=t)
14        t+=a[j].y2-a[j].y1+1;
15    printf("%d\n",t<=m?t:0);
16  }
17 }

```

3. (Reactivi - **) Într-un laborator de analize chimice se utilizează $N \leq 8000$ reactivi. Se știe că, pentru a evita accidente sau deprecierea reactivilor, aceștia trebuie să fie stocați în condiții de mediu speciale. Mai exact, pentru fiecare reactiv x se precizează intervalul de temperatură $[\min_x, \max_x]$ ($-100 \leq \min_x, \max_x \leq 100$) în care trebuie să se încadreze temperatura de stocare a acestuia. Reactivii vor fi plasați în frigider. Orice frigider are un dispozitiv cu ajutorul căruia putem stabili temperatura (constantă) care va fi în interiorul celui frigider (exprimată într-un număr întreg de grade Celsius). Scrieți un program care să determine numărul minim de frigider necesare pentru stocarea reactivilor chimici, știind că un frigider poate conține un număr nelimitat de reactivi.

Fișierul de intrare *react.in* conține pe prima linie numărul natural N , care reprezintă numărul de reactivi, iar pe fiecare dintre următoarele N linii se află *min max* (două numere întregi separate printr-un spațiu); numerele de pe linia $x+1$ reprezintă temperatura minimă, respectiv temperatura maximă de stocare a reactivului x . Fișierul de ieșire *react.out* va conține o singură linie pe care este scris numărul minim de frigider necesar.

Exemplu:

react.in		react.out
4		3
2 5		
5 7		
10 20		
30 40		

(O.J.I.2004, clasa a IX-a)

Soluție:

Se sortează intervalele de temperatură după temperatura maximă, iar intervalele care se intersectează se vor „uni” într-un singur interval. Numărul de intervale rămase va reprezenta răspunsul.

```

1 type interval=record x,y:integer; end;
2 var n,i,j,nr:integer; t:interval;
3 a:array[1..8000] of interval;
4 begin
5   assign(input,'react.in'); reset(input); read(n);
6   for i:=1 to n do read(a[i].x,a[i].y);
7   for i:=1 to n do
8     for j:=i+1 to n do
9       if a[i].y>a[j].y then begin
10        t:=a[i]; a[i]:=a[j]; a[j]:=t; end;
11   nr:=1; j:=1;
12   for i:=2 to n do
13     if a[j].y<a[i].x then begin inc(nr); j:=i; end;
14   assign(output,'react.out'); rewrite(output); writeln(nr);
15 end.

```

```

1 #include <stdio.h>
2 typedef struct { int x,y; } interval;
3 interval a[8000]; int n;
4 void main() {
5   int i,j,nr;
6   freopen("react.in","r",stdin); scanf("%d",&n);
7   for (i=0;i<n;i++) scanf("%d%d",&a[i].x,&a[i].y);
8   for (i=0;i<n;i++)
9     for (j=i+1;j<n;j++)
10      if (a[i].y>a[j].y)
11        { t=a[i]; a[i]=a[j]; a[j]=t; }
12   for (nr=1,i=1,j=0;i<n;i++)
13     if (a[j].y<a[i].x) { nr++; j=i; }
14   freopen("react.out","w",stdout); printf("%d\n",nr);
15 }

```

4. (Litere - ***) Se consideră un șir de litere mari ale alfabetului latin de lungime cel mult 10000. Acest șir se află pe prima linie a unei matrice pătratică A , având dimensiunea egală cu lungimea șirului. Următoarea linie a matricei A se obține prin permutarea la stânga cu o poziție a elementelor de pe prima linie. Aceeași operație se aplică și pentru următoarele linii: linia i se obține prin permutarea la stânga cu o poziție a elementelor de pe linia $i-1$. În continuare, se generează o matrice B care conține liniile matricei A , ordonate lexicografic. De exemplu, dacă șirul de caractere considerat este TEST, atunci matricea A va avea următoarea structură:

```
TEST
ESTT
STTE
TTES
```

Ordonând lexicografic liniile matricei A , se obține matricea B :

```
ESTT
STTE
TEST
TTES
```

Se cunosc caracterele de pe ultima coloană a matricei B și se cere determinarea caracterelor de pe prima linie a acesteia.

Fișierul de intrare *litere.in* conține o singură linie pe care se află caracterele de pe ultima coloană a matricei B .

Fișierul de ieșire *litere.out* va conține o singură linie pe care se vor afla caracterele de pe prima linie a matricei B .

Exemplu:

	litere.in		litere.out
TETS		ESTT	

Soluție:

Algoritmul de rezolvare al acestei probleme constă în următorii pași:

- se determină frecvența de apariție a literelor, ceea ce permite obținerea primei coloane a matricei (prima coloană conține întotdeauna literele ordonate),
- se construiește un șir de corespondențe A între literele de pe prima coloană și cele de pe ultima, ținând cont de faptul că celei de-a i -a apariții a literei c de pe prima coloană îi corespunde cea de-a i -a apariție a literei c de pe ultima.
- pentru determinarea șirului cerut, se scriu literele corespunzătoare pozițiilor $A[1]$, $A[A[1]]$, $A[A[A[1]]]$, etc.; pentru aceasta se folosește un indice i care, după fiecare pas va primi valoarea $A[i]$.

```
1 var n,i,j:integer; c:char;
2   nr:array['A'..'Z'] of integer;
3   A:array[1..10000] of integer;
4   x:array[1..10000] of char;
5 begin
6   assign(input,'litere.in'); reset(input);
7   while not seekeof(input) do begin inc(n); read(x[n]); end;
8   for i:=1 to n do inc(nr[x[i]]);
9   for c:='B' to 'Z' do inc(nr[c],nr[pred(c)]);
```

```
10 for i:=n downto 1 do begin A[nr[x[i]]]:=i; dec(nr[x[i]]); end;
11 assign(output,'litere.out'); rewrite(output); j:=1;
12 for i:=1 to n do begin
13   write(x[A[j]]); j:=A[j];
14 end;writeln;
15 end.
```

```
1 #include <stdio.h>
2 #include <string.h>
3 char x[10001];int n,nr[256],A[10000];
4 void main() {
5   int i,j;
6   freopen("litere.in","r",stdin);
7   scanf("%s\n",x); n=strlen(x);
8   for (i=0;i<n;i++) nr[x[i]]++;
9   for (i='A';i<='Z';i++) nr[i]+=nr[i-1];
10  for (i=n-1;i>=0;i--) A[--nr[x[i]]]=i;
11  freopen("litere.out","w",stdout);
12  for (i=j=0;i<n;i++,j=A[j]) printf("%c",x[A[j]]);
13  putchar('\n');
14 }
```

5. (Criptare - *) Mircea și Vasilică vor să-și trimită mesaje pe care alții să nu le înțeleagă. Au citit ei despre spioni și despre modalități de a scrie mesaje și, în final, au imaginat un mod de criptare a unui mesaj care folosește "cuvânt cheie" (le-a plăcut lor denumirea acesta). Alegându-și un cuvânt cheie (cuvântul cheie are minimum 5 și maximum 20 de caractere), format numai din litere mici distincte, ei numără literele acestuia și împart mesajul în grupe de lungime egală cu numărul de litere ale cuvântului cheie, și le așază una sub alta. Desigur, se poate întâmpla ca ultima grupă să fie incompletă, așa că o completează cu spații. Apoi numerotează literele cuvântului cheie în ordinea apariției lor în alfabetul englezesc. În final, rescriu mesajul astfel: coloana de sub litera numerotată cu 1, urmată de coloana de sub litera numerotată cu 2, etc. înlocuind totodată și spațiile cu caracterul '*' (asterisc). Spre exemplu:

cuvântul cheie	criptam
mesaj de criptat	Incercam sa lucram cu coduri si criptari.
cuvântul cheie	criptam are 7 litere
numerotate	2635714
deoarece, avem, în ordine	abcdefghijklmnopqrstuvwxy
	1 2 3 4 5 6 7
împărțire în grupe	Incerca m sa lu cram cu coduri si cri ptari.
codificare	2635714
	Incerca
	m*sa*lu
	cram*cu
	*coduri
	*si*cri
	ptari.*
mesaj criptat	clcrr.lmc**pcsaioiauuui*eamd*rn*rcstr**uci
	col1 col2 col3 col4 col5 col6 col7

Fiind date un cuvânt cheie și un mesaj criptat, decodificați mesajul trimis de Mircea pentru Vasilică.

Fișierul de intrare *criptare.in* conține pe prima linie mesajul criptat, iar pe linia a doua cuvântul cheie. Lungimea mesajului este de minimum 20 și maximum 1000 caractere.

Fișierul de intrare *criptare.out* conține pe prima linie mesajul decriptat.

Exemplu:

criptare.in	criptare.out
clcr. lmc**pcsaoliaauuii*eamd*rn*rcstr**uci	Inceram sa lucram cu
criptam	coduri si criptari.
	(O.N.I.2003, clasa a IX-a)

Soluție:

Pentru decriptarea mesajului, se utilizează algoritmul de criptare descris în sens invers:

- se împarte mesajul în fragmente;
- se așează într-o matrice de caractere;
- se determină, în cuvântul cheie, ordinea caracterelor din alfabet;
- se afișează pe rând coloanele corespunzătoare, în ordine.

```

1 var n,m,i,j,t:integer;
2   s1,s2:array[1..1000] of char;
3   a:array[0..200,1..20] of char;
4 begin
5   assign(input,'criptare.in'); reset(input);
6   while not seekoln(input) do begin
7     inc(n);
8     read(s1[n]);
9   end;
10  readln;
11  while not seekoln(input) do begin inc(m); read(s2[m]); end;
12  for i:=1 to m do begin
13    t:=0;
14    for j:=1 to m do
15      if s2[j]<s2[i] then inc(t);
16    for j:=(n div m)*t to (n div m)*(t+1)-1 do
17      a[j mod (n div m),i]:=s1[j+1];
18  end;
19  assign(output,'criptare.out'); rewrite(output);
20  for i:=0 to (n div m)-1 do
21    for j:=1 to m do
22      if (a[i,j]<>'') and (a[i,j]<>#0) then
23        write(a[i,j]);
24      else if (a[i,j]=#0) then begin
25        i:=n div m; j:=m+1;
26      end else
27        write(' ');
28  writeln;
29 end.
```

```

1 #include <stdio.h>
2 #include <string.h>
3 int n,m; char s1[1001],s2[1001],a[201][21];
4 void main() {
5   int i,j,t;
6   freopen("criptare.in","r",stdin);
7   scanf("%s\n%s",s1,s2); n=strlen(s1); m=strlen(s2);
8   for (i=0;i<m;i++){
9     for (t=j=0;j<m;j++) if (s2[j]<s2[i])t++;
10    for(j=(n/m)*t;j<(n/m)*(t+1);j++)
11      a[j%(n/m)][i]=s1[j];
12  }
13  freopen("criptare.out","w",stdout);
14  for (i=0;i<n/m;i++)
15    for (j=0;j<m;j++)
16      if (a[i][j]&&a[i][j]!='') putchar(a[i][j]);
17      else if (!a[i][j]) i=n/m,j=m;
18      else putchar(' ');
19  putchar('\n');
20 }
```

6. (Codificare - *) Doru și Stelică, doi elevi neastâmpărați, vor să comunice între ei în timpul orelor, prin texte codificate. Pentru acest lucru ei ajung la ideea că nu trebuie să se complice prea mult pentru a putea decodifica ușor textele. Astfel ei codifică fiecare cuvânt schimbând între ele prima cu ultima literă, apoi a doua literă cu penultima și așa mai departe (spre exemplu, cuvântul *acest* va fi codificat prin *tseca*). Pentru a simplifica comunicarea ei vor utiliza numai texte formate din litere mici ale alfabetului englezesc și spații. Spațiile nu vor fi codificate, ele fiind lăsate pe aceleași poziții în text. Două cuvinte vor fi separate printr-unul sau prin mai multe spații. De la tastatură se dă un text cu maximum 80 de caractere, iar textul codificat va fi afișat de către programul vostru pe ecran.

Exemplu:

intrare	iesire
mircea a spart un geam	aecrim a traps nu maeg

Soluție:

Se împarte textul în cuvinte, și se inversează fiecare cuvânt.

```

1 var n,i,j,st,dr:integer; s:string[80];
2 begin
3   read(s);s:=s+' '; n:=length(s);
4   st:=0; dr:=0;
5   for i:=1 to n do
6     if s[i]=' ' then begin
7       j:=dr;
8       while (j>=st) and (j>0) do begin
9         write(s[j]);
10        dec(j);
11      end;
12      write(' ');
13      st:=0;
14      dr:=0;
15    end
```

```

16 else begin
17   if st=0 then st:=i;
18   dr:=i;
19 end;
20 writeln;
21 end.

```

```

1 #include <stdio.h>
2 #include <string.h>
3 char s[81]; int n;
4 void main() {
5   int i, j, st, dr;
6   gets(s); strcat(s, " "); n=strlen(s);
7   for (st=dr=-1, i=0; i<n; i++)
8     if (s[i]==' '){
9       for (j=dr; j>=st&&j>=0; j--)
10        putchar(s[j]);
11       st=dr=-1; putchar(' ');
12     }
13   else {
14     if (st===-1) st=i;
15     dr=i; }
16   putchar('\n');
17 }

```

7. (Șir - *) Să considerăm următorul șir:

a, b, ba, bab, babba, babbabab, ...

Determinați care este cel de-al n -lea termen al șirului, $n \leq 20$.

Fișierul de intrare *sir.in* conține o singură linie pe care se află numărul natural n .

Fișierul de ieșire *sir.out* va conține o singură linie pe care se află al n -lea termen din șir.

Exemplu:

sir.in	bab	sir.out
4		(O.J.I.2003, clasa a VII-a)

Soluție:

Șirul de caractere este cunoscut ca "șirul de caractere Fibonacci", deoarece este format în mod asemănător celui matematic, Fibonacci. Șirul de lungime n va avea lungime $Fib(n)$.

```

1 var n,i,j,k:integer;
2   s1,s2,s3:array[1..15000] of char;
3 begin
4   s1[1]:='a'; s2[1]:='b';
5   assign(input,'sir.in'); reset(input); read(n);
6   for i:=3 to n do begin
7     s3:=s2;
8     j:=1; while s3[j]<>#0 do inc(j);
9     k:=1; while s1[k]<>#0 do begin s3[j]:=s1[k]; inc(j); inc(k); end;
10    s1:=s2; s2:=s3;
11  end;

```

```

12 assign(output,'sir.out'); rewrite(output);
13 i:=1; while s3[i]<>#0 do begin write(s3[i]); inc(i); end;
14 writeln;
15 end.

```

```

1 #include <stdio.h>
2 #include <string.h>
3 int n; char s1[15000]="a", s2[15000]="b", s3[15000];
4 void main() {
5   int i;
6   freopen("sir.in","r",stdin); scanf("%d",&n);
7   for (i=3; i<=n; i++) {
8     strcpy(s3,s2);
9     strcat(s3,s1);
10    strcpy(s1,s2);
11    strcpy(s2,s3);
12   }
13   freopen("sir.out","w",stdout); printf("%s\n",s3);
14 }

```

8. (Paranteze - **) Considerăm șiruri formate din paranteze de două tipuri: paranteze rotunde și paranteze drepte. Parantezele se codifică în felul următor: paranteză rotundă deschisă cu 0, paranteză rotundă închisă cu 1, paranteză dreaptă deschisă cu 2, paranteză dreaptă închisă cu 3. Spre deosebire de convenția uzuală din matematică, aici pot exista și paranteze rotunde incluse în paranteze drepte și paranteze drepte incluse în paranteze rotunde. Nu putem asocia unei paranteze rotunde deschise o paranteză dreaptă închisă sau viceversa. Să se decidă dacă un astfel de șir este corect construit, în sensul că putem asocia corect două câte două paranteze de fiecare tip.

Fișierul text *par.in* conține pe prima linie numărul $n \leq 10$ (numărul de șiruri ale testului). Apoi pe fiecare din liniile 2... $n+1$ se află numerele:

$L \ c_1 \ c_2 \ \dots \ c_L$

Numărul natural $L \leq 500$ reprezintă lungimea unui șir de paranteze codificat conform enunțului. Valorile c_1, c_2, \dots, c_L reprezintă codurile respective. Toate numerele sunt despărțite prin câte un spațiu.

În fișierul text *par.out* se vor scrie n linii. Pe câte o linie va fi scris câte un mesaj. Pe fiecare linie se va scrie unul dintre mesajele 'Da', respectiv 'Nu', reprezentând rezultatele verificării corectitudinii șirurilor. Ordinea lor corespunde ordinii șirurilor din fișierul de intrare. Exemplu:

par.in		par.out
8		Da
6 0 2 3 1 0 1		Da
6 2 0 1 0 1 3		Nu
4 0 2 1 3		Nu
6 2 0 0 3 1 1		Nu
6 2 2 0 1 1 3		Da
6 2 0 0 1 3 0		Da
10 2 2 0 1 0 1 3 3 0 1		
10 0 0 0 1 1 1 0 2 3 1		

(O.J.I. 2003, clasa a VII-a)

Soluție:

Pentru a verifica dacă o paranteză este validă, se va folosi o structură de date numită *stivă*. Se parcurge șirul caracter cu caracter, iar când caracterul curent este o paranteză deschisă, se introduce în vârful stivei; când caracterul curent este o paranteză închisă, se verifică dacă paranteza deschisă din vârful stivei este de același tip cu cea curentă, se elimină din vârful stivei, altfel se scrie "Nu". La sfârșitul parcurgerii șirului se afișează "Nu" dacă stiva nu este goală.

```
1 type stiva=record
2   nr:integer; inf:array[1..500] of byte;
3 end;
4 var n,i,j,l,x:integer; ok:boolean; st:stiva;
5 begin
6   assign(input,'par.in'); reset(input);
7   assign(output,'par.out'); rewrite(output);
8   readln(n);
9   for i:=1 to n do begin
10    read(l); ok:=true; st.nr:=0;
11    for j:=1 to l do begin
12     read(x);
13     if (x=0) or (x=2) then begin
14      inc(st.nr); st.inf[st.nr]:=x;
15     end else begin
16      if (st.nr=0) or (x<>st.inf[st.nr]+1) then ok:=false;
17      dec(st.nr);
18     end;
19    end;
20    if (ok) and (st.nr<>0) then ok:=false;
21    if ok then writeln('Da') else writeln('Nu');
22  end;
23 end.
```

```
1 #include <stdio.h>
2 typedef struct { int nr; char inf[500]; } stiva;
3 stiva st; int n,l;
4 void main() {
5   int i,j,x,ok;
6   freopen("par.in","r",stdin);
7   freopen("par.out","w",stdout);
8   scanf("%d",&n);
9   for (i=0;i<n;i++) {
10    scanf("%d",&l); ok=1; st.nr=0;
11    for (j=0;j<l;j++){
12     scanf("%d",&x);
13     if (x==0 || x==2) st.inf[st.nr++]=x;
14     else {
15      if (!st.nr || x!=st.inf[st.nr-1]+1) ok=0;
16      st.nr--;
17     }
18    }
19    if (ok && st.nr) ok=0;
20    printf("%s\n",ok?"Da":"Nu");
21  }
22 }
```

9. (Text - *) Se dă un text format din mai multe cuvinte separate prin anumite caractere. Aceste caractere de separare pot fi: spațiu, punct, virgulă, punct și virgulă sau două puncte. Să se determine toate perechile de cuvinte ce pot fi obținute prin anagramare, unul din altul.

Fișierul de intrare *text.in* conține (pe mai multe linii, eventual) textul de parcurs, iar fișierul de ieșire *text.out* trebuie să conțină pe linii separate perechi de cuvinte din text cu proprietatea că unul poate fi anagramarea celui alt. Nu se va face distincție între litere mari și litere mici, iar un cuvânt poate avea maximum 10 litere, iar tot textul, cel mult 100 de cuvinte.

Exemplu:

text.in	text.out
rac acar.topor,car;ropot:arac bca cab bac.	rac car
	arac acar
	ropot topor
	bca cab
	bca bac
	cab bac

Soluție:

Se împarte textul în cuvinte, apoi se ia fiecare pereche de cuvinte și se verifică dacă sunt anagrame.

```
1 var n,m,i,j,k,st,dr:integer; c:char;
2   s:array[1..10000] of char;
3   w:array[1..100] of string[10];
4   nr:array[#0..#255] of byte;
5 begin
6   assign(input,'text.in'); reset(input);
7   while not eof(input) do begin inc(n); read(s[n]); end;
8   inc(n); s[n]:=' '; st:=0; dr:=0;
9   for i:=1 to n do
10    if (s[i]=' ') or (s[i]='.') or (s[i]=';') or (s[i]=':') or (s[i]='<') then begin
11     if st=0 then continue;
12     inc(m);
13     for j:=st to dr do w[m]:=w[m]+s[j];
14     st:=0;
15     dr:=0;
16   end else begin
17     if st=0 then st:=i;
18     dr:=i; end;
19   assign(output,'text.out'); rewrite(output);
20   for i:=1 to n do
21    for j:=i+1 to n do
22     if length(w[i])=length(w[j]) then begin
23      fillchar(nr,sizeof(nr),0);
24      for k:=1 to length(w[i]) do inc(nr[w[i][k]]);
25      for k:=1 to length(w[j]) do dec(nr[w[j][k]]);
26      for c:=#0 to #255 do if nr[c]>0 then break;
27      if c=#255 then writeln(w[i],' ',w[j]);
28     end;
29   end.
```

```

1 #include <stdio.h>
2 #include <string.h>
3 int n; char s[10000], w[101][11], *p, nr[256];
4 void main() {
5     int i, j, k;
6     freopen("text.in", "r", stdin); gets(s);
7     p = strtok(s, " .,:;");
8     for (; p; p = strtok(NULL, " .,:;")) strcpy(w[n++], p);
9     freopen("text.out", "w", stdout);
10    for (i=0; i<n; i++)
11        for (j=i+1; j<n; j++)
12            if (strlen(w[i]) == strlen(w[j]))
13            {
14                memset(nr, 0, sizeof(nr));
15                for (k=0; w[i][k]; k++) nr[w[i][k]]++;
16                for (k=0; w[j][k]; k++) nr[w[j][k]]--;
17                for (k=0; k<256; k++) if (nr[k]) break;
18                if (k==256) printf("%s %s\n", w[i], w[j]);
19            }
20 }

```

10. (Rational - **) Numerele raționale pozitive sunt acele numere care sunt egale cu m/n , cu m și n numere naturale, prime între ele, ≤ 32.000 . Există un mod foarte interesant de a obține toate numerele raționale: pentru început, se consideră trei numere: $0/1$ adică 0, $1/1$ adică 1 și $1/0$ adică un fel de infinit. Pornind de la acestea putem obține un nou șir de numere raționale combinând două numere consecutive. Se aleg cele două numere. Fie ele de tipul a/b și c/d . Noul număr rezultat este $(a+c)/(b+d)$ și se pune între cele două. De exemplu primului șir îi urmează: $0/1$, $(0+1)/(1+1)$, adică $1/2$, $1/1$, $(1+1)/(1+0)$ adică $2/1$, $1/0$. Prin acest procedeu se poate genera un fel de arbore:

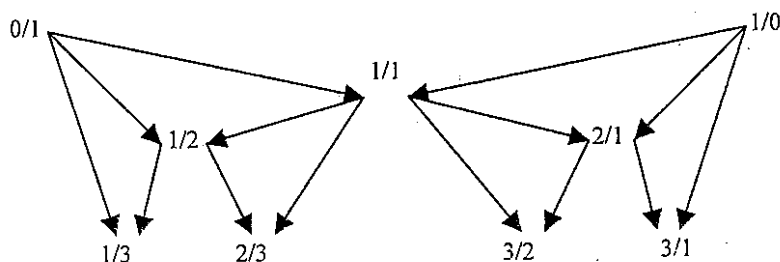
Pe nivelul 0 se află: $0/1$ și $1/0$;

Pe nivelul 1 se află: $0/1$, $1/1$, $1/0$;

Pe nivelul 2 se află: $0/1$, $1/2$, $1/1$, $2/1$, $1/0$;

Pe nivelul 3 se află: $0/1$, $1/3$, $1/2$, $2/3$, $1/1$, $3/2$, $2/1$, $3/1$, $1/0$.

Vi se cere să identificați drumul pe care trebuie să îl străbătați în arbore, pentru a ajunge la o anumită fracție. Traseul începe din vârf ($1/1$) și merge la fiecare pas, fie spre stânga, fie spre dreapta. De exemplu, ca să dăm de $2/3$ mergem prima oară în stânga și a doua oară în dreapta.



Din fișierul *rational.in* se citesc două numere m și n , separate printr-un spațiu alb reprezentând numărul pe care îl căutăm (m/n).

În fișierul *rational.out* se vor scrie pașii pe care îi urmăm ("S" de la stânga sau "D" de la dreapta) câte unul pe linie.

Exemplu:

	<i>rational.in</i>		<i>rational.out</i>
1			S
4			S
			S

Soluție:

Pentru fiecare fracție din arbore, la stânga se află numai fracții mai mici, iar la dreapta numai fracții mai mari. Astfel, de fiecare dată se compară fracția căutată cu fracția curentă și se decide în ce direcție se va merge.

```

1 type fracție=record a,b:double; end;
2 var tmp,st,dr,mij:fracție;
3 begin
4     assign(input,'rational.in'); reset(input);
5     readln(tmp.a,tmp.b);
6     assign(output,'rational.out'); rewrite(output);
7     st.a:=0; st.b:=1;
8     mij.a:=1; mij.b:=1;
9     dr.a:=1; dr.b:=0;
10    while tmp.a/tmp.b<>mij.a/mij.b do
11        if tmp.a/tmp.b<mij.a/mij.b then begin
12            writeln('S');
13            dr:=mij; mij.a:=mij.a+st.a; mij.b:=mij.b+st.b;
14        end
15        else begin
16            writeln('D');
17            st:=mij; mij.a:=mij.a+dr.a; mij.b:=mij.b+dr.b;
18        end;
19    end.

```

```

1 #include <stdio.h>
2 typedef struct { double a,b; } fracție;
3 fracție st,mij,dr,tmp;
4 void main() {
5     freopen("rational.in", "r", stdin);
6     scanf("%lf\n%lf\n",&tmp.a,&tmp.b);
7     freopen("rational.out", "w", stdout);
8     st.a=0; st.b=1;
9     mij.a=1; mij.b=1;
10    dr.a=1; dr.b=0;
11    while (tmp.a/tmp.b!=mij.a/mij.b)
12        if (tmp.a/tmp.b<mij.a/mij.b) {
13            printf("S\n");
14            dr=mij; mij.a+=st.a; mij.b+=st.b;
15        } else {
16            printf("D\n");
17            st=mij; mij.a+=dr.a; mij.b+=dr.b;
18        }
19 }

```


11. (Permutări - ***) Pentru o mulțime oarecare de numere întregi cu N elemente se poate defini o permutare a acestora ca fiind o variantă de a așeza elementele mulțimii. De exemplu, pentru mulțimea $M=\{4,12,81\}$ vom avea permutările:

- 1) { 4 , 12 , 81 }
- 2) { 4 , 81 , 12 }
- 3) { 12 , 4 , 81 }
- 4) { 12 , 81 , 4 }
- 5) { 81 , 4 , 12 }
- 6) { 81 , 12 , 4 }

În grupul format de toate permutările unei mulțimi se poate defini o ordine a acestora astfel încât elementele aflate pe prima poziție să fie în ordine crescătoare, elementele de pe a doua poziție să fie în ordine crescătoare pentru permutările care au primul element identic, etc. așa cum se vede în figură. O astfel de ordine se numește *ordine lexicografică*. Permutărilor așezate în ordine lexicografică li se pot asocia numere de ordine: în cazul exemplului de mai sus, de la 1) la 6).

Dându-se o mulțime de numere întregi și o permutare a ei, concepeți un program care determină numărul de ordine al permutării în ordine lexicografică.

Fișierul de intrare *perm.in* conține pe prima linie N =numărul de elemente al mulțimii, $1 \leq N \leq 1.000$, pe a doua linie elementele mulțimii, în ordine crescătoare, $1 \leq m_i \leq 2.000.000.000$, pe a treia linie elementele permutării (elementele $m_1, m_2, m_3, \dots, m_N$, dar în altă ordine)

Fișierul de ieșire *perm.out* va conține numărul permutării în ordine lexicografică.

Exemplu:	<i>perm.in</i>	17	<i>perm.out</i>
	4		
	1 2 3 4		
	3 4 1 2		

(<http://infoarena.devnet.ro>)

Soluție: Se transformă permutarea dată cu o permutare echivalentă, dar cu elemente cu valori între 1 și N , înlocuind fiecare termen cu poziția lui în mulțime. Apoi, se parcurge permutarea element cu element și se adună la rezultat câte permutări de lungime curentă există, se elimină primul element și se renumerează permutarea. Ar trebui implementate numere mari pentru păstrarea rezultatului; lăsăm această implementare ca exercițiu pentru cititor.

```

1 var n,i,j:integer; fact,rez:longint;
2   m,p:array[1..1000] of longint;
3 begin
4   assign(input,'perm.in'); reset(input);
5   readln(n);
6   for i:=1 to n do read(m[i]);
7   for i:=1 to n do begin
8     read(p[i]);
9     for j:=1 to n do
10      if m[j]=p[i] then break;
11     p[i]:=j-1;
12   end;
```

```

13 fact:=1; rez:=1;
14 for i:=2 to n-1 do fact:=fact*i;
15 for i:=1 to n do begin
16   inc(rez,p[i]*fact);
17   for j:=i+1 to n do
18     if p[j]>p[i] then dec(p[j]);
19   if i<n then fact:=fact div (n-i);
20 end;
21 assign(output,'perm.out'); rewrite(output);
22 writeln(rez);
23 end.
```

```

1 #include <stdio.h>
2 int n; long m[1000],p[1000];
3 void main() {
4   int i,j; long fact,rez=1;
5   freopen("perm.in","r",stdin);
6   scanf("%d",&n);
7   for (i=0;i<n;i++) scanf("%ld",m+i);
8   for (i=0;i<n;i++) {
9     scanf("%ld",p+i);
10    for (j=0;j<n;j++) if (m[j]==p[i]) break;
11    p[i]=j;
12  }
13  for(fact=i=1;i<n;i++) fact*=i;
14  for (i=0;i<n;i++) {
15    rez+=p[i]*fact;
16    for(j=i+1;j<n;j++)
17      if (p[j]>p[i]) p[j]--;
18    if (i+1<n) fact/= (n-i-1);
19  }
20  freopen("perm.out","w",stdout);
21  printf("%ld\n",rez);
22 }
```

12. (Numere - *) Pe fiecare linie a unui fișier text se găsesc mai multe succesiuni de 0 și 1 (fiecare succesiune începe cu 1), fiecare linie reprezintă un număr în baza 2. Dându-se un astfel de fișier, realizați un program care să determine cel mai mare număr par (afișat în format zecimal) care există în fișier - se garantează că există cel puțin un număr par în fișier!

Datele se citesc din fișierul *numere.in* care conține mai multe linii formate din succesiuni de 0 și 1, separate prin câte un spațiu.

Prima linie a fișierului *numere.out* conține un număr care va reprezenta cel mai mare număr par care există în fișierul de intrare.

Exemplu:	<i>numere.in</i>	6	<i>numere.out</i>
	1 1 1		
	1 0		
	1 0 1 1		
	1 1 0 1		
	1 1 0		

(<http://infoarena.devnet.ro>)

Soluție:

Se transformă fiecare număr binar din baza 2 în baza 10 și se verifică dacă este par, caz în care se actualizează soluția.

```
1 type binar=record n:integer; cif:array[1..100] of byte; end;
2 var s:string; i,x,nr,max:integer; b:binar;
3 begin
4   assign(input,'numere.in');reset(input);
5   while not seekeof(input) do begin
6     readln(s);
7     x:=length(s);
8     b.n:=0;
9     i:=1;
10    while ((s[i]='0')or(s[i]='1'))and(i<=x) do begin
11      inc(b.n);
12      b.cif[b.n]:=ord(s[i])-ord('0');
13      inc(i,2);
14    end;
15    nr:=0;
16    for i:=1 to b.n do
17      inc(nr,b.cif[i]*(1 shl (b.n-i)));
18    if (nr mod 2=0)and(max<nr) then max:=nr;
19  end;
20  assign(output,'numere.out');rewrite(output);
21  writeln(max);
22 end;
```

```
1 #include <stdio.h>
2 #include <string.h>
3 typedef struct { int n; char cif[100]; } binar;
4 char s[256];
5 void main(void) {
6   int i,nr,max=0; binar b;
7   freopen("numere.in", "r", stdin);
8   while (!feof(stdin)) {
9     memset(s, 0, sizeof(s));
10    fgets(s, 255, stdin);
11    for (b.n=i=0;s[i]!='0' || s[i]!='1';i+=2)
12      b.cif[b.n++]=s[i]-'0';
13    for (nr=i=0;i<b.n;i++)
14      nr+=b.cif[i]*(1<<(b.n-i-1));
15    if (nr%2==0 && max<nr) max=nr;
16  }
17  freopen("numere.out", "w", stdout);
18  printf("%d\n",max);
19 }
```

13. (Mulțimi - *) Se consideră două mulțimi cu elemente naturale din intervalul [1,30000]. Să se determine reuniunea celor două mulțimi.

Datele de intrare se citesc din fișierul text *multimi.in*, ce conține pe prima linie cardinalul $c1 \leq 30000$ al primei mulțimi, pe următoarele $c1$ linii se află elementele primei mulțimi (câte un element pe fiecare linie), pe următoarea linie se află

cardinalul $c2 \leq 30000$ al celei de-a doua mulțimi, iar pe următoarele $c2$ linii se găsesc elementele celei de-a doua mulțimi (câte un element pe fiecare linie). Datele de ieșire se tipăresc în fișierul text *multimi.out*, care are următoarea structură: pe prima linie se va scrie un număr natural $c3$, reprezentând cardinalul reuniunii celor două mulțimi, iar pe următoarele $c3$ linii se vor scrie elementele reuniunii, în ordine crescătoare.

Exemplu:

<i>multimi.in</i>	<i>multimi.out</i>
3	4
1	1
4	2
2	4
2	10
10	
2	

(<http://infoarena.devnet.ro>)

Soluție:

Deoarece mulțimile au doar elemente din intervalul [1, 30.000], reținem într-un vector cu elemente dintr-un tip logic dacă un element aparține sau nu mulțimii. Astfel, procesul de reuniune se realizează ușor.

```
1 type multime=record n:integer; elm:array[1..30000] of boolean; end;
2 var a,b:multime; i,x:integer;
3 begin
4   assign(input,'multimi.in'); reset(input);
5   readln(a.n);
6   for i:=1 to a.n do begin readln(x); a.elm[x]:=true; end;
7   readln(b.n);
8   for i:=1 to b.n do begin readln(x); b.elm[x]:=true; end;
9   a.n:=0;
10  for i:=1 to 30000 do begin
11    a.elm[i]:=a.elm[i] or b.elm[i];
12    if a.elm[i] then inc(a.n);
13  end;
14  assign(output,'multimi.out'); rewrite(output);
15  writeln(a.n);
16  for i:=1 to 30000 do
17    if a.elm[i] then writeln(i);
18 end;
```

```
1 #include <stdio.h>
2 typedef struct { int n; char elm[30001]; } multime;
3 multime a,b;
4 void main() {
5   int i,x;
6   freopen("multimi.in", "r", stdin);
7   scanf("%d",&a.n);
8   for (i=0;i<a.n;i++){ scanf("%d",&x); a.elm[x]=1; }
9   scanf("%d",&b.n);
10  for (i=0;i<b.n;i++){ scanf("%d",&x); b.elm[x]=1; }
11  for (a.n=0,i=1;i<=30000;i++) {
12    a.elm[i]=b.elm[i]; a.n+=a.elm[i];
13  }
```

```

14 freopen("multimi.out", "w", stdout);
15 printf("%d\n", a.n);
16 for (i=1; i<=30000; i++)
17     if (a.elm[i]) printf("%d\n", i);
18 }

```

14. (Bin - *) Să se determine toate numerele scrise în baza 2, care au $n \leq 10$ cifre, sunt prime și conțin exact $p \leq n$ cifre de 1.

Datele de intrare se citesc din fișierul text *bin.in*, ce conține pe prima linie două numere naturale n și p (n – numărul de cifre, p – numărul de cifre „1”)

Datele de ieșire (numerele ce respectă condițiile din enunț) se tipăresc în fișierul text *bin.out* pe linii separate, în ordine crescătoare. Pe fiecare linie se scriu două numere separate prin spațiu: numărul în baza 2 și numărul transformat în baza 10.

Exemplu:

	<i>bin.in</i>		<i>bin.out</i>
4	2	0011 3	
		0101 5	

(<http://infoarena.devnet.ro>)

Soluție:

Se generează toate numerele binare de lungime n și se verifică care sunt prime și au p cifre de 1.

```

1 type binar=record n:integer; cif:array[1..10] of byte; end;
2 var n,p,i,j,k:integer; ok:boolean; b:binar;
3 begin
4   assign(input, 'bin.in'); reset(input);
5   assign(output, 'bin.out'); rewrite(output);
6   readln(n,p);
7   for i:=2 to (1 shl n) do begin
8     fillchar(b, sizeof(binar), 0);
9     j:=i; while j>0 do begin inc(b.n); b.cif[b.n]:=j mod 2; j:=j div 2; end;
10    k:=0;
11    for j:=1 to b.n do inc(k, b.cif[j]);
12    if k<>p then continue;
13    ok:=true;
14    for j:=2 to trunc(sqrt(i)) do
15      if i mod j=0 then begin
16        ok:=false; break;
17      end;
18    if not ok then continue;
19    for j:=n downto 1 do write(b.cif[j]);
20    writeln(' ', i);
21  end;
22 end.

```

```

1 #include <stdio.h>
2 #include <string.h>
3 typedef struct { int n; char cif[10]; } binar;
4 int n,p;
5 binar b;

```

```

6 void main() {
7   int i,j,k,ok;
8   freopen("bin.in", "r", stdin);
9   freopen("bin.out", "w", stdout);
10  scanf("%d%d", &n, &p);
11  for (i=2; i<=(1<<n); i++){
12    memset(&b, 0, sizeof(binar));
13    for (j=i; j>0; j/=2) b.cif[b.n++]=j%2;
14    for (j=k=0; j<b.n; j++) k+=b.cif[j];
15    if (k!=p) continue;
16    for (ok=1, j=2; j*j<=i; j++)
17      if (i%j==0) { ok=0; break; }
18    if (!ok) continue;
19    for (j=n-1; j>=0; j--) printf("%d", b.cif[j]);
20    printf(" %d\n", i);
21  }
22 }

```

15. (Concert - ***) Un grup de $N \leq 1.000$ artiști a primit invitații să cânte la un concert. Fiecărui artist i s-a trimis o invitație pe care acesta era convocat să cânte între orele A și $B \leq 2.000.000.000$. Oricare doi artiști au intervalele diferite. Însă organizatorii acestui eveniment au greșit invitațiile și timpurile în care ar fi trebuit să cânte anumiți artiști se suprapuneau. Artiștii, fiind renumiți, nu acceptă să cânte decât singuri. De asemenea ei cer despăgubiri dacă li se cere să înceapă să cânte după ora $A+1$ sau să termine înainte de ora B . Un anumit artist poate să cânte doar între orele $A+1$ și B de pe invitație. Unele invitații pot fi anulate, așa că unii artiști pot să nu cânte deloc. Cunoscându-se intervalele între care poate să cânte fiecare artist, profitul pe care îl aduce fiecare artist pe unitatea de timp și despăgubirile cerute în ambele cazuri menționate mai sus, voi trebuie să aflați care este suma maximă care poate fi obținută în concertul respectiv, și cum poate fi ea obținută. Profitul adus de fiecare artist sau despăgubirile cerute de fiecare dintre ei nu vor trece de valoarea 500. Despăgubirile se plătesc o dată, nu pe unitate de timp.

Pe prima linie a fișierului *concert.in* se va afla N . Pe următoarele N linii ale fișierului se vor găsi câte cinci numere separate prin câte un spațiu. Ele reprezintă A, B , profitul pe unitatea de timp și despăgubirile, în cazul în care artistul nu începe să cânte în $A+1$ și respectiv, dacă nu termină de cântat în B .

Pe prima linie a fișierului *concert.out* se va afla suma maximă care poate fi obținută din concert.

Exemplu:	<i>concert.in</i>		<i>concert.out</i>
5		189	
0	5 10 10 15		
3	8 8 8 20		
4	10 12 4 10		
8	16 7 4 7		
12	20 10 5 50		

(<http://infoarena.devnet.ro>)

Soluție:

Se sortează intervalele după capătul dreapta crescător, iar în caz de egalitate, după capătul stâng crescător. Se calculează în $C[i]$ suma maximă care poate fi obținută

din concert, dacă artistul i cântă. Această valoare se calculează în funcție de valorile $C[j]$ ($j < i$), luând în considerare două cazuri: dacă artistul j termină de cântat înainte să înceapă artistul i sau invers.

```
1 type interval=record a,b,c,p1,p2:longint; end;
2 var n,i,j,t,rez:longint;
3     tmp:interval;
4     E:array[1..1000] of interval;
5     C:array[1..1000] of longint;
6 begin
7   assign(input,'concert.in'); reset(input);
8   readln(n);
9   for i:=1 to n do read(E[i].a,E[i].b,E[i].c,E[i].p1,E[i].p2);
10  for i:=1 to n do
11    for j:=i+1 to n do
12      if (E[i].b>E[j].b) or ((E[i].b=E[j].b) and (E[i].a>E[j].a)) then begin
13        tmp:=E[i];
14        E[i]:=E[j];
15        E[j]:=tmp;
16      end;
17    for i:=1 to n do begin
18      C[i]:=(E[i].b-E[i].a)*E[i].c;
19      for j:=1 to i-1 do
20        if E[j].b<E[i].a then begin
21          t:=C[j]+(E[i].b-E[i].a)*E[i].c;
22          if C[i]<t then C[i]:=t;
23        end
24        else begin
25          t:=C[j]-E[j].p2-(E[j].b-E[i].a)*E[j].c+(E[i].b-E[i].a)*E[i].c;
26          if C[i]<t then C[i]:=t;
27          t:=C[j]-E[i].p1+(E[i].b-E[j].b)*E[i].c;
28          if C[i]<t then C[i]:=t;
29        end;
30      if rez<C[i] then rez:=C[i];
31    end;
32  assign(output,'concert.out');
33  rewrite(output);
34  writeln(rez);
35 end.
```

```
1 #include <stdio.h>
2 typedef struct { long a,b,c,p1,p2; } interval;
3 long n,C[1000],rez;
4 interval E[1000],tmp;
5 void main(){
6   long i,j,t;
7   freopen("concert.in","r",stdin);
8   scanf("%ld",&n);
9   for (i=0;i<n;i++)
10     scanf("%ld%ld%ld%ld",&E[i].a,&E[i].b,&E[i].c,&E[i].p1,&E[i].p2);
11   for (i=0;i<n;i++)
12     for (j=i+1;j<n;j++)
13       if (E[i].b>E[j].b || (E[i].b==E[j].b && E[i].a>E[j].a))
14         { tmp=E[i]; E[i]=E[j]; E[j]=tmp; }
```

```
15 for (i=0;i<n;i++) {
16   C[i]=(E[i].b-E[i].a)*E[i].c;
17   for (j=0;j<i;j++)
18     if (E[j].b<E[i].a) {
19       t=C[j]+(E[i].b-E[i].a)*E[i].c;
20       if (C[i]<t) C[i]=t;
21     }
22   else {
23     t=C[j]-E[j].p2-(E[j].b-E[i].a)*E[j].c+
24     (E[i].b-E[i].a)*E[i].c;
25     if (C[i]<t) C[i]=t;
26     t=C[j]-E[i].p1+(E[i].b-E[j].b)*E[i].c;
27     if (C[i]<t) C[i]=t;
28   }
29   if (rez<C[i]) rez=C[i];
30 }
31 freopen("concert.out","w",stdout);
32 printf("%ld\n",rez);
33 }
```

1.3.2 Probleme propuse

1. (Cuvinte - **) Considerăm un text ce conține doar litere mari și mici ale alfabetului englezesc și cratime ('-'), iar cuvintele sunt separate prin spații, punct, virgulă, new-line, semnul întrebării și semnul exclamării. Se cere determinarea cuvintelor distincte din text și afișarea acestora în ordinea crescătoare a lungimii. Cuvintele cu aceeași lungime se vor tipări în ordine alfabetică. Numărul de cuvinte distincte din text este cel mult 500, iar lungimea unui cuvânt nu depășește 100 de caractere. Datele de intrare se citesc din fișierul text *cuvinte.in*, ce conține textul scris pe un număr arbitrar de linii. Datele de ieșire se tipăresc în fișierul text *cuvinte.out*, care va conține lista cuvintelor, în ordinea crescătoare a lungimii acestora, câte unul pe linie.

Exemplu: *cuvinte.in*
 Ana n-are mere. Gîna are
 pere , prune,
 caise .

cuvinte.out
 *
 Ana
 are
 Gîna
 mere
 pere
 caise
 n-are
 prune

(<http://infoarena.devnet.ro>)

2. (Matrice - *) Pentru un număr natural n , să se construiască o matrice de forma:

```
1 2 3 4 ... n
2 1 2 3 ... n-1
3 2 1 2 ... n-2
. . . . .
n n-1 ... 1
```

Datele se citesc din fișierul *matrice.in*, iar fișierul *matrice.out* va conține matricea construită pentru numărul natural $n \leq 1.000$.

Exemplu: *matrice.in* *matrice.out*

5	1	2	3	4	5
	2	1	2	3	4
	3	2	1	2	3
	4	3	2	1	2
	5	4	3	2	1

(<http://infoarena.devnet.ro>)

3. (*Pascal - ***) Ion a învățat la ora de matematică despre triunghiul lui *Pascal*. Fiecare rând din acest triunghi are primul și ultimul element egal cu 0. Un element din triunghi se calculează ca fiind suma celor 2 elemente exact deasupra acestuia. Rândurile sunt numerotate de la 0, deci, spre exemplu, rândul 2 conține: 1 2 1. Este un fapt binecunoscut că valoarea elementului j (cu indexarea elementelor de la 0) de pe linia i se poate calcula și cu ajutorul formulei: $i! / ((i-j)! * j!)$. (Prin $i!$ se înțelege produsul $1 * 2 * \dots * i$). Ajuțați-l pe Ion să calculeze câte numere de pe rândul $R \leq 5.000.000$ sunt divizibile la $D \leq 6$.

Pe prima linie a fișierului de intrare *pascal.in* se găsesc numerele R și D , iar pe prima linie a fișierului de ieșire *pascal.out* se va găsi numărul cerut.

Exemplu: *pascal.in* *pascal.out*

4 2	3
-----	---

(<http://infoarena.devnet.ro>)

4. (*Text - ***) Dezamăgit de rezultatele sale la ultimul concurs, Paftenie a renunțat la programare și s-a concentrat strict asupra muncii laborioase, dar care implică mai puțin efort intelectual. De această dată, el primește un text de cel mult 1.000.000 de caractere și trebuie să calculeze lungimea medie a cuvintelor textului, un cuvânt fiind definit ca o secvență continuă maximală de caractere ale alfabetului englezesc ('a' .. 'z', 'A' .. 'Z'). Definim lungimea medie = (lungimea totală a cuvintelor textului) / (numărul de cuvinte al textului).

Scrieți un program care îi rezolvă problema lui Paftenie. Fișierul de intrare *text.in* conține textul dat. Fișierul de ieșire *text.out* va conține pe prima linie un singur întreg, reprezentând partea întreagă a lungimii medii a cuvintelor textului.

Exemplu: *text.in* *text.out*

- Lasa-ma in pace, ca am invatat azi noapte toata ziua!	3
---	---

(<http://infoarena.devnet.ro>)

5. (*Cod secret - **) Simpatie mare între Ionel și Mărioara, doi elevi veniți în tabără la Gălăciuc...! Pentru a scăpa de indiscreția colegilor, cei doi hotărăsc să-și trimită mesaje, unul altuia, folosind o metodă simplă de criptare: textul de criptat se scrie pe o foaie, aranjând literele cuvintelor într-un tablou având câte 5 caractere pe fiecare linie. Spațiul dintre cuvinte este și el caracter. Textul astfel aranjat pe un număr suficient de linii pentru a încapa, se citește pe coloane, de sus în jos și de la

stânga la dreapta. În locul spațiilor dintre cuvinte se pun puncte. Tot puncte se pun și la sfârșitul textului, atâtea câte spații libere sunt rămase la sfârșitul textului "pus" în tablou.

Pentru textul: „Te aștept după cina la ora 8”.

Se va aranja:

1	2	3	4	5
T	e		a	s
t	e	P	t	
d	u	P	a	
c	i	N	a	
l	a		o	r
a		8		

se va codifica: *Ttdclaeuia..ppn.8ataao.s...r*.

Decodificarea mesajului se va face invers codificării.

Ajuțați-i, realizând un program care să codifice și să decodifice mesajele celor doi copii. Pentru diferențierea mesajelor ce trebuie codificate, de cele care trebuie decodificate, primul caracter al mesajului va fi 'C' sau 'c' pentru codificare, respectiv 'D' sau 'd' pentru decodificare. Aceste caractere, vor fi lipite de prima literă din textul mesajului.

Exemplu: *intrare* *iesire*

CAm un mar	A.mm.aurn.
dTaGia.aubllcaaa.r.c.	Tabara la Galaciuc

(O.N.I.2002, clasa a VI-a)

6. (*Arhivare - ***) Se propune următoarea schemă de arhivare/dezarhivare pentru texte care nu conțin caractere numerice. La arhivare, orice caracter nealfabetic al textului este copiat direct în textul arhivat. Un cuvânt este copiat în textul arhivat doar dacă este la prima apariție a lui și, de asemenea, este adăugat la sfârșitul unei liste de cuvinte. În aparițiile ulterioare cuvântul va fi înlocuit cu numărul de ordine din listă respectivă (numerotarea pozițiilor în listă începe de la 1).

Scrieți un program care citește din fișierul *arhiva.in* un text format din mai multe linii, determină tipul operației (arhivare/dezarhivare) și scrie în fișierul *arhiva.out* textul arhivat/dezarhivat. În text există maximum 2000 de cuvinte, iar cuvintele au cel mult 20 de caractere și liniile textului au cel mult 80 de caractere. Nu se va face deosebire între literele mari și cele mici.

Exemplu: *arhiva.in* *arhiva.out*

Mult mai mult din putin cat mai putin, mai-mai ca te pune pe ganduri cat mai mult.	Mult mai 1 din putin cat 2 4, 2-2 ca te pune pe ganduri 5 2 1.
--	---

Observații:

- În exemplul de mai sus, "mai-mai" reprezintă două cuvinte.
- Un text se consideră că trebuie dezarhivat dacă el conține cel puțin o cifră.

- Caracterele alfabetice sunt: A..Z, a..z;
 - Lista de cuvinte generată în exemplul precedent este:
- 1) mult, 2) mai, 3) din, 4) puțin, 5) cat, 6) ca, 7) te, 8) pune, 9) pe, 10) gânduri.

(O.N.I.2002, clasa a VII-a)

7. (**Cut&Paste** - ***) Să considerăm un fișier constituit din $N \leq 100.000$ linii, pe fiecare linie fiind câte un număr natural. Inițial, pe linia 1 se află valoarea 1, pe linia 2, valoarea 2, ș.a.m.d., pe linia N se află valoarea N . Fișierul inițial a fost modificat cu ajutorul unui editor de texte, prin efectuarea a $M \leq 1.000$ operații de tip *cut&paste*. O operație *cut&paste* constă în selectarea unui grup de linii consecutive, eliminarea lor din poziția inițială și inserarea lor într-o altă poziție în document. Scrieți un program care, pentru o secvență de operații *cut&paste* dată, determină conținutul primelor 10 linii din fișierul obținut după efectuarea operațiilor.

Fișierul de intrare *cp.in* conține pe prima linie numărul de linii din fișier și numărul de operații efectuate, iar pe următoarele M linii câte trei numere: l_i și $l_f \leq N$ – linia de început și linia de sfârșit ale celui de al k -lea grup de linii selectat și $unde_k \leq N - (l_f - l_i + 1)$ – linia după care se va insera grupul de linii specificat. Dacă $unde_k = 0$, deducem că inserarea se va face la începutul documentului.

Fișierul de ieșire *cp.out* va conține 10 linii reprezentând numerele scrise pe primele 10 linii din fișierul obținut după efectuarea operațiilor.

Exemplu:	<i>cp.in</i>	<i>cp.out</i>
1000 6	801	
3 7 4	802	
1 100 57	803	
50 60 200	804	
63 70 500	101	
1 800 4	102	
7 77 98	36	
	37	
	38	
	39	

(O.N.I.2002, clasa a VIII-a)

8. (**Fib** - **) Considerăm șirul lui *Fibonacci*: 0,1,1,2,3,5,8,13,21,...

Fiind dat un număr natural ($n \in \mathbb{N}$), scrieți acest număr sub formă de sumă de elemente neconsecutive din șirul *Fibonacci*, fiecare element putând să apară cel mult o dată, astfel încât numărul de termeni ai sumei să fie minim.

Din fișierul de intrare *fib.in* se citește de pe prima linie numărul natural n . Acesta poate avea maximum 80 de cifre.

În fișierul de ieșire *fib.out* se vor afișa termeni ai șirului *Fibonacci*, câte unul pe linie, a căror sumă este n .

Exemplu:	<i>fib.in</i>	<i>fib.out</i>
20	2	
	5	
	13	

(O.N.I.2000, clasa a IX-a)

9. (**Pentagon** - *) În urma bombardamentelor din 11 septembrie 2001, clădirea Pentagonului a suferit daune la unul din pereții clădirii. Imaginea codificată a peretelui avariat se reprezintă sub forma unei matrice cu m linii și n coloane ($m, n \leq 200$) ca în figura de mai jos:

1110000111	unde	1 reprezintă zid intact
1100001111		0 zid avariat
1000000011		
1111101111		
1110000111		

Sumele alocate pentru refacerea Pentagonului vor fi donate celor care vor ajuta americanii să refacă această clădire prin plasarea, pe verticală, a unor blocuri de înălțimi $k, k=1, \dots, m$, și lățime 1, în locurile avariate.

Pentru o structură dată a unui perete din clădirea Pentagonului, determinați numărul minim al blocurilor, de înălțimi $k=1, k=2, \dots, k=m$, necesare refacerii clădirii.

Fișierul de intrare *pentagon.in* conține pe prima linie dimensiunile m și n ale peretelui clădirii, iar pe următoarele m linii câte o secvență de caractere 1 sau 0 de lungime n . Fișierul *pentagon.out* va conține pe câte o linie, ordonate crescător după k , secvențe:

$k \text{ nr}$ - unde k este înălțimea blocului,

- iar nr este numărul de blocuri de înălțime k , separate prin câte un spațiu.

Exemplu:	<i>pentagon.in</i>	<i>pentagon.out</i>
5 10		1 7
1110000111		2 1
1100001111		3 2
1000000011		5 1
1111101111		
1110000111		

(O.N.I.2003, clasa a IX-a)

10. (**Perle** - ****) Granița nu se trece ușor. Asta pentru că Balaurul Arhirel (mare pasionat de informatică) nu lasă pe nimeni să treacă decât după ce răspunde la niște întrebări... În acea țară există trei tipuri de perle normale (le vom nota cu 1, 2 și 3) și trei tipuri de perle magice (le vom nota cu A, B și C). Perlele magice sunt deosebite prin faptul că se pot transforma în alte perle (una sau mai multe, normale sau magice). Perla magică de tipul A se poate transforma în orice perlă normală (una singură). Perla magică de tipul B se poate transforma într-o perlă normală de tipul 2 și una magică de tipul B, sau într-o perlă normală de tipul 1, una magică de tipul A, una normală de tipul 3, una magică de tipul A și una magică de tipul C. Perla magică de tipul C se poate transforma într-o perlă normală de tipul 2 sau într-o perlă normală de tipul 3, una magică de tipul B și una magică de tipul C sau într-o perlă normală de tipul 1, una normală de tipul 2 și una magică de tipul A.

Ca să rezumăm cele de mai sus putem scrie:

A ->	1	2	3
B ->	2B	1A3AC	
C ->	2	3BC	12A

Balaurul Arhirel ne lasă la început să ne alegem o perlă magică (una singură), iar apoi folosind numai transformările de mai sus trebuie să obținem un anumit șir de perle normale. Când o perlă magică se transformă, perlele din stânga și din dreapta ei rămân la fel (și în aceeași ordine). De asemenea, ordinea perlelor rezultate din transformare este chiar cea prezentată mai sus.

De exemplu, dacă balaurul ne cere să facem șirul de perle 21132123, putem alege o perlă magică de tipul B și următorul șir de transformări: B → 2B → 21A3AC → 21A3A12A → 21132123. Întrucât Balaurul nu are prea multă răbdare, el nu ne cere decât să spunem dacă se poate sau nu obține șirul respectiv de perle.

Să se determine pentru fiecare șir de intrare dacă se poate obține prin transformările de mai sus sau nu (alegând orice primă perlă magică, la fiecare șir).

Fișierul de intrare *perle.in* are următoarea structură: pe prima linie, numărul $N \leq 10$, reprezentând numărul de șiruri din fișierul de intrare; urmează N linii, a i -a linie dintre cele N descrie șirul i , printr-o succesiune de numere naturale despărțite de câte un spațiu. Primul număr reprezintă lungimea șirului $L_i \leq 10.000$, iar următoarele L_i numere sunt tipurile de perle normale, în ordine, de la stânga la dreapta.

Fișierul *perle.out* va conține N linii. Pe linia i se va scrie un singur număr 1 sau 0 (1 dacă se poate obține șirul respectiv (al i -lea) și 0 dacă nu se poate).

Exemplu:

<i>perle.in</i>		<i>perle.out</i>
3		1
8 2 1 1 3 2 1 2 3		0
2 2 2		1
1 3		

(O.J.I.2004, clasa a X-a)

11. (*Șir - ***) Se consideră următorul șir, construit astfel încât fiecare element al lui, cu excepția primului, se obține din cel precedent: 1, 11, 21, 1211, 111221, ...

Termenii din șir sunt numerotați începând cu 1.

Fiind dat $n \leq 35$, un număr natural, să se determine cel de-al n -lea termen din șirul dat.

Din fișierul text *sir.in* se citește numărul natural n .

Pe prima linie a fișierului text *sir.out* se va scrie al n -lea termen al șirului.

Exemplu:

<i>sir.in</i>		<i>sir.out</i>
5		111221

(O.J.I.2002, clasa a VIII-a)

12. (*Șablon - **) Gigel și Vasilică imaginează un mod de a transmite mesaje pe care nimeni să nu le poată descifra. Mesajul este ascuns într-un text care are N linii și pe fiecare linie sunt exact $N \leq 50$ caractere – litere mari ale alfabetului englezesc, cifre, semne de punctuație și caracterul spațiu. Decodificarea se face cu ajutorul unui șablon, de aceleași dimensiuni ca și textul, care are câteva găuri. Suprapunând șablonul peste text rămân vizibile câteva caractere. Acestea se citesc în ordinea liniilor, de sus în jos, iar pe aceeași linie de la stânga la dreapta. Apoi hârtia cu textul se rotește spre stânga, în sens trigonometric, cu 90° , șablonul rămânând fix. Alte caractere devin vizibile și acestea se citesc în același mod. Operația se repetă

de încă două ori (rotire cu 180° , respectiv cu 270°), până când textul ajunge, printr-o nouă rotație cu 90° , din nou în poziția inițială. Din păcate, șablonul pentru codificare/decodificare s-a pierdut. În schimb a rămas la Gigel mesajul inițial, iar la Vasilică a ajuns textul care conține mesajul. Să se reconstituie șablonul care a fost folosit la codificare. Prin rotirea textului nici una din găuri nu se va suprapune peste nici una din pozițiile ocupate de o gaură în pozițiile precedente ale textului.

Fișierul de intrare *sablon.in* conține, pe prima linie, mesajul inițial. Mesajul are maximum 1000 caractere și se încheie cu un caracter diferit de spațiu. Pe linia a doua a fișierului de intrare se găsește valoarea N . Următoarele N linii conțin textul care ascunde mesajul.

Fișierul de ieșire *sablon.out* conține N linii a câte N caractere. Caracterele sunt 'O' (pentru reprezentarea unei găuri) și 'X'.

Exemplu:

<i>sablon.in</i>		<i>sablon.out</i>
CODIFICARE CU SABLON		XXXXXXXXXX
10		XXOXXXXXXXX
ABCDEFGHIJ		OXXXXXXXXXX
IJKLMNOP		XXOXXXXXXXX
QRSTUVWXYZ		XOXXXXXXXXX
YZAIBCRDEF		XXXXXXXXXXXX
GFHIJKLMNI		XXXXXXXXXXXX
AJKLMNOPSQ		XXXXXXXXXXXX
RSTOUV WXY		XXXXXXXXXXXX
ZBABCDEFGR		XXXXXXXXXXXX
HIJKNLMCNO		XXXXXXXXXXXX
PQLRS TUVW		XXXXXXXXXXXX

(O.N.I.2004, clasa a IX-a)

13. (*Seti - **) Cercetătorii care lucrează la programul SETI au recepționat două transmisii de date foarte ciudate, date care ar putea veni din partea unor civilizații extraterestre. Primul set de date este format din 10 caractere distincte, date în ordinea lor lexicografică, ce formează alfabetul extraterestru. A doua transmisie conține cuvinte din exact 4 caractere.

Cercetătorii trebuie să ordoneze lexicografic cuvintele primite în a doua transmisie (conform alfabetului extraterestru).

Fișierul de intrare *seti.in* conține pe prima linie cele 10 caractere ale alfabetului, iar pe fiecare din următoarele linii, câte un cuvânt. În fișier nu sunt mai mult de 200.000 de cuvinte, iar caracterele sunt literele mici ale alfabetului englezesc.

Fișierul de ieșire *seti.out* va conține cuvintele ordonate, câte unul pe linie.

Exemplu:

<i>seti.in</i>		<i>seti.out</i>
abcdefghij		aaaa
aaaa		aabc
fgaa		fgaa
aabc		iihf
iihf		

(O.N.I. 2003, clasa a IX-a)

14. (Text - **) Vasile lucrează intens la un editor de texte. Un text este format din unul sau mai multe paragrafe. Orice paragraf se termină cu Enter și oricare două cuvinte consecutive din același paragraf sunt separate prin spații (unul sau mai multe). În funcție de modul de setare a paginii, numărul maxim de caractere care încap în pagină pe o linie este unic determinat (Max). Funcția pe care Vasile trebuie să o implementeze acum este alinierea în pagină a fiecărui paragraf din text la stânga și la dreapta. Pentru aceasta el va trebui să împartă fiecare paragraf în linii separate de lungime Max (fiecare linie terminată cu Enter). Împărțirea se realizează punând numărul maxim posibil de cuvinte pe fiecare linie, fără împărțirea cuvintelor în silabe. Pentru aliniere stânga-dreapta, el trebuie să repartizeze spații în mod uniform între cuvintele de pe fiecare linie, astfel încât ultimul caracter de pe linie să fie diferit de spațiu, iar numărul total de caractere de pe linie să fie egal cu Max. Excepție face numai ultima linie din paragraf, care rămâne aliniată la stânga (cuvintele fiind separate printr-un singur spațiu, chiar dacă linia nu este plină). În general, este puțin probabil ca alinierea să fie realizabilă prin plasarea aceluiași număr de spații între oricare două cuvinte consecutive de pe linie. Vasile consideră că este mai elegant ca, dacă între unele cuvinte consecutive trebuie plasat un spațiu în plus față de alte perechi de cuvinte consecutive, acestea să fie plasate la începutul liniei. Scrieți un program care să citească lungimea unei linii și textul dat și care să alinieze textul la stânga și la dreapta. Lungimea maximă a oricărui cuvânt din text este 25 de caractere și nu depășește Max. Lungimea unui paragraf nu depășește 1000 de caractere.

Fișierul de intrare *text.in* conține pe prima linie $Max \leq 1.000$, lungimea maximă a unui rând. Pe următoarele linii este scris textul.

Fișierul de ieșire *text.out* conține textul aliniat stânga-dreapta.

<i>Exemplu:</i>	<i>text.in</i>	<i>text.out</i>
20	Vasile are multe	bomboane bune.
Vasile are multe bomboane bune.		

(O.J.I.2003, clasa a IX-a)

15. (Cod - **)** Principala misiune a unei expediții științifice este de a studia evoluția vieții pe o planetă nou descoperită. În urma studiilor efectuate, cercetătorii au asociat fiecărui organism viu descoperit pe acea planetă un cod caracteristic. Codul caracteristic este un număr natural de maximum 200 de cifre zecimale nenule. De asemenea, cercetătorii au observat că pentru orice organism viu de pe planetă, codurile caracteristice ale strămoșilor săi pe scara evoluției se pot obține prin ștergerea unor cifre din codul caracteristic al organismului respectiv, iar un organism este cu atât mai evoluat cu cât codul său caracteristic are o valoare mai mare. Date fiind codurile caracteristice ale două organisme vii diferite, scrieți un program care să determine codul caracteristic al celui mai evoluat strămoș comun al lor.

Fișierul de intrare *cod.in* conține n - codul caracteristic al primului organism pe prima linie și m - codul caracteristic al celui de-al doilea organism pe a doua linie.

Fișierul de ieșire *cod.out* conține pe prima linie codul celui mai evoluat strămoș comun al lui n și m .

<i>Exemplu:</i>	<i>cod.in</i>
7145	
847835	

75

cod.out

(O.J.I.2002, clasa a X-a)

16. (Palindrom - *)** Un palindrom este un șir simetric, adică un șir care este la fel citit de la stânga la dreapta, cât și de la dreapta la stânga. Scrieți un program, care determină pentru un șir dat, numărul minim de caractere care trebuie inserate în șir pentru ca acesta să devină palindrom.

În fișierul de intrare *pal.in* va fi pe prima linie numărul de caractere al șirului, iar pe a doua linie, șirul care va fi format doar din litere mici și mari ale alfabetului.

În fișierul de ieșire *pal.out* se va scrie numărul minim de caractere care trebuie inserate.

<i>Exemplu:</i>	<i>pal.in</i>	<i>pal.out</i>
5		
Abxbd		

2

17. (Program - *)** Atunci când este analizată o sursă scrisă într-un anumit limbaj de programare, este util să verificăm dacă există instrucțiuni care, cu siguranță nu vor fi executate niciodată. Dacă există, atunci acestea ar putea reprezenta o eroare în program. Să considerăm un limbaj de programare simplu, care admite următoarele instrucțiuni:

EXECUTA - execuția programului continuă cu următoarea instrucțiune;

SALT n - execuția programului continuă cu cea de a n -a instrucțiune;

SALT n SAU m - execuția programului continuă cu cea de a n -a sau cea de a m -a instrucțiune. Execuția unui program începe întotdeauna cu prima instrucțiune.

Scrieți un program care să determine numărul de instrucțiuni care nu vor fi niciodată executate. Numărul de instrucțiuni este cel mult egal cu 10000. Instrucțiunile sunt numerotate începând cu 1.

Fișierul de intrare *program.in* conține mai multe instrucțiuni, câte o instrucțiune pe linie. Ultima linie din fișier conține (ca marcaj de sfârșit) caracterul '?

Fișierul de ieșire *program.out* conține o singură linie pe care se află numărul de instrucțiuni care nu vor fi niciodată executate.

<i>Exemplu:</i>	<i>program.in</i>	<i>program.out</i>
EXECUTA		
SALT 4 SAU 6		
EXECUTA		
SALT 3		
EXECUTA		
SALT 8		
EXECUTA		
EXECUTA		

2

18. (Romane - **) În reprezentarea cu cifre romane se utilizează simbolurile I, V, X, L, C care reprezintă respectiv valorile 1, 5, 10, 50 și 100. Pentru a reprezenta alte valori, aceste simboluri se alătură și se adună. De exemplu valoarea 3 se reprezintă prin III, iar valoarea 73 prin LXXIII. Excepțiile de la regulă apar la numerele care au una dintre cifre 4 sau 9: 4 se scrie IV, 9 se scrie IX, 40 se scrie

XL, 90 se scrie XC, adică valoarea mai mică se scrie în fața valorii mai mari și se scade din ea. Astfel, reprezentările cu cifre romane pentru 24, 39, 44, 49 și 94 sunt respectiv XXIV, XXXIX, XLIV, XLIX, XCIV. În multe cărți prefața are paginile numerotate cu cifre romane, începând cu I pentru prima pagină. Scrieți un program care să determine câte caractere I, V, X, L, C sunt utilizate pentru a numerota cele $n < 400$ pagini din prefață.

Fișierul de intrare *romane.in* conține pe prima linie un singur întreg reprezentând numărul de pagini din prefață.

Fișierul de ieșire *romane.out* va conține o singură linie pe care se vor afla cinci numere naturale separate prin câte un spațiu reprezentând, în ordine, numerele de caractere I, V, X, L, C necesare.

Exemplu *romane.in* | *romane.out*
20 | 28 10 14 0 0
(<http://campion.edu.ro>)

19. (*Secvreg* - ***) Să considerăm secvențe constituite numai din paranteze rotunde și paranteze pătrate, adică din caracterele () [].

Prin definiție, o secvență de paranteze este *regulată* dacă se obține aplicând următoarele reguli:

1. () și [] sunt secvențe regulate.

2. Dacă A este regulată, atunci (A) și [A] sunt secvențe regulate.

3. Dacă A și B sunt regulate, atunci AB este secvență regulată.

De exemplu, () () [] și [(())] sunt regulate, în timp ce][sau [(] sau (]) nu sunt regulate. Se consideră o secvență de paranteze. La fiecare pas se înserează o paranteză (rotundă sau pătrată, deschisă sau închisă) la începutul sau la sfârșitul secvenței. Scrieți un program care să determine, după fiecare pas, lungimea celei mai scurte subsecvențe regulate (formată din caractere consecutive ale secvenței) care conține paranteza inserată la pasul respectiv.

Fișierul de intrare *secvreg.in* conține pe prima linie secvența inițială de paranteze. Lungimea secvenței inițiale de paranteze va fi $\leq 100\,000$. Pe cea de a doua linie a fișierului de intrare se află un număr natural $N \leq 100\,000$ care reprezintă numărul de pași. Fiecare dintre următoarele N linii conține un număr natural A și un caracter C separate printr-un singur spațiu. Dacă A este 0 (zero) atunci caracterul C este inserat la începutul secvenței; dacă A este 1 (unu) atunci caracterul C este inserat la sfârșitul secvenței.

Fișierul de ieșire *secvreg.out* conține N linii. Pe cea de a i-a linie va fi afișată lungimea celei mai scurte subsecvențe regulate (formată din caractere consecutive ale secvenței) care conține paranteza inserată la pasul i. Dacă nu există o astfel de subsecvență, pe linia respectivă veți afișa 0.

Exemplu *secvreg.in* | *secvreg.out*
[()] | 0
3 | 2
0) | 6
0 (|
0 (|

20. (*Tunel* - **) Ion stă la intrarea în tunel, iar Vasile stă la ieșire. Fiecare își notează numărul de înregistrare al fiecărei mașini care trece pe lângă el și trimite aceste informații patrului de Poliție care așteaptă pe drum, ceva mai la vale. Utilizând informațiile oferite de Ion și Vasile, Poliția poate determina dacă anumiți șoferi au făcut depășiri în tunel (ceea ce este strict interzis). Presupunem că în tunel mașinile nu se pot opri.

Scrieți un program care să determine numărul de șoferi despre care Poliția poate afirma cu siguranță că au făcut o depășire în tunel.

Fișierul de intrare *tunel.in* conține $2N+1$ linii. Pe prima linie se află un număr natural $N \leq 1.000$, care reprezintă numărul de mașini care au trecut prin tunel. Pe următoarele N linii se află numerele de înregistrare ale mașinilor care au intrat în tunel, în ordinea intrării. Pe următoarele N linii se află numerele de înregistrare ale mașinilor care au ieșit din tunel în ordinea ieșirii. Numerele de înregistrare ale mașinilor au cel puțin 6 și cel mult 8 caractere care pot fi numai litere mari ale alfabetului englezesc și cifre.

Fișierul de ieșire *tunel.out* conține o singură linie pe care se află numărul de șoferi care vor primi amendă (cei despre care Poliția poate afirma cu siguranță că au efectuat depășiri în tunel).

Exemplu *tunel.in* | *tunel.out*
5 | 3
ZG5080K
PU305A
RI604B
ZG206A
ZG232ZF
PU305A
ZG232ZF
ZG206A
ZG5080K
RI604B
(<http://campion.edu.ro>)

21. (*Circular* - ***) Se spune că șirul y_1, y_2, \dots, y_n este o permutare circulară cu p poziții a șirului x_1, x_2, \dots, x_n dacă $y_1 = x_{p+1}, y_2 = x_{p+2}, \dots, y_n = x_{p+n}$, unde indicii k, cu $k > n$ se referă la elementul de indice k-n. Pentru două șiruri date determinați dacă al doilea este o permutare circulară a primului șir.

Pe prima linie a fișierului de intrare *circular.in* este scris numărul natural $n \leq 5000$. Pe liniile următoare sunt două șiruri de caractere de lungime n, formate numai din litere mari ale alfabetului latin.

Pe prima linie a fișierului *circular.out* se va scrie cel mai mic număr natural p pentru care șirul de pe linia a treia este o permutare circulară cu p poziții a șirului de pe linia a doua, sau numărul -1 dacă nu avem o permutare circulară.

Exemplu: *circular.in* | *circular.out*
10 | 7
ABCBAABBAB
BABABCBAAB
(<http://campion.edu.ro>)

22. (*Super* - **) Numerele romane, folosite pe vremea lui Midas exprimau dificil numere mari, de ordinul milioane. De aceea, Midas a fost nevoit sa inventeze numere noi, pe care le-a numit numere super-romane. Numerele super-romane verifică regulile obișnuite de formare a numerelor romane. Să considerăm simbolurile obișnuite romane și reprezentarea lor zecimală:

I = 1 V = 5 X = 10 L = 50 C = 100 D = 500 M = 1000

Simbolurile I, X, C, M pot fi scrise consecutiv în număr de maximum 3. Celelalte simboluri nu se pot repeta consecutiv. Simbolurile sunt scrise fără spații între ele, în ordinea descrescătoare a valorilor:

CCLXVIII = 100+100+50+10+5+1+1+1 = 268

Uneori însă, un simbol I, X, C sau M este plasat înaintea unuia dintre cele două simboluri de valori imediat superioare (I înaintea de V sau X; X înaintea de L sau C, etc). În acest caz, valoarea simbolului mai mic este scăzută din valoarea simbolului pe care îl precedă:

IV = 5-1 = 4 IX = 10-1 = 9 XL = 50-10 = 40

Numere compuse însă, ca XD, IC sau XM nu sunt corecte, deoarece simbolul din față are o valoare mult prea mică față de a celui care urmează. Astfel:

- pentru XD (490 scris greșit) se folosește reprezentarea CDXC
- pentru IC (99 scris greșit) se folosește XCIX
- pentru XM (990 scris greșit) se folosește CMXC

Din nefericire, numere mari, cum ar fi 10000 s-ar reprezenta MMMMMMMMMM, deci foarte dificil. De aceea, tabela numerelor romane se extinde:

I = 1 L = 50 M = 1000 R = 50000 U = 1000000 N = 50000000
V = 5 C = 100 P = 5000 S = 100000 B = 5000000 Y = 100000000
X = 10 D = 500 Q = 10000 T = 500000 W = 10000000 Z = 500000000

În acest fel, păstrând regulile obișnuite de scriere, se pot reprezenta și numere de ordinul sutelor de milioane. Scrieți un program care citește un număr natural dintr-un fișier și îl convertește în numărul echivalent în formă super-romană.

Pe prima linie a fișierului de intrare *super.in* se găsește numărul natural $n \leq 2000000000$.

Pe prima linie a fișierului de ieșire *super.out* se găsește numărul n în formă super-romană.

Exemplu: *super.in* | *super.out*
12345678 | WUOSSQRPDCLXXVIII

(<http://campion.edu.ro>)

23. (*Cifre* - **) Fie $N \leq 1.000.000.000$ un număr natural nenul. Scriind în ordine, unul după altul, toate numerele naturale de la 1 la N obținem o secvență de cifre. De exemplu, pentru $N=22$ obținem: 12345678910111213141516171819202122.

Scrieți un program care să determine numărul de cifre dintr-o astfel de secvență.

Fișierul de intrare *cifre.in* conține o singură linie pe care se află numărul natural N .

Fișierul de ieșire *cifre.out* conține o singură linie pe care se află numărul de cifre determinat.

Exemplu: *cifre.in* | *cifre.out*
15 | 21

(<http://campion.edu.ro>)

24. (*Parola* - ***) Într-un seif se află niște documente pe care trebuie să le extrageți. Problema este că seiful este prevăzut cu un terminal care necesită introducerea unei parole pentru a-l putea deschide. La accesarea seifului, pe ecranul terminalului este afișat un cuvânt cheie format din litere mici ale alfabetului englezesc. Parola este dată de cea mai mică rotație la stânga (în ordine lexicografică) a cuvântului cheie.

Fișierul de intrare *parola.in* conține pe prima linie un șir de caractere format din litere mici ale alfabetului englezesc. Lungimea șirului din fișierul de intrare este un număr întreg cuprins între 1 și 100.000.

Fișierul de ieșire *parola.out* trebuie să conțină un singur număr care reprezintă numărul necesar de deplasări circulare la stânga ale șirului din fișierul de intrare pentru a obține parola de acces cerută. Dacă există mai multe soluții, va fi aleasă cea care necesită un număr minim de deplasări circulare la stânga.

Exemplu: *parola.in* | *parola.out*
mississippi | 10

25. (*Frază* - ***) Un strămoș elf a scris un text pentru ca posteritatea să cunoască acțiunile sale. Din nefericire, el nu a separat cuvintele între ele. Elfii cunosc cuvintele folosite de strămoși și doresc acum să reconstituie fraza.

Fișierul de intrare *fraza.in* conține, pe prima linie, textul scris de strămoșul lor. Textul strămoșesc conține cel mult 30000 de litere. Pe următoarea linie se află numărul n al cuvintelor din limba strămoșilor pe care le cunosc elfii. Numărul cuvintelor strămoșești este cuprins între 1 și 255. Fiecare dintre următoarele n linii va conține câte un cuvânt din această limbă. Fiecare cuvânt strămoșesc conține cel mult 255 de litere.

Fișierul de ieșire *fraza.out* trebuie să conțină pe prima linie numărul k al cuvintelor din textul strămoșesc reconstituit. Fiecare dintre următoarele k linii va conține cuvintele textului, câte unul pe fiecare linie. Ordinea acestor linii trebuie să fie cea a apariției cuvintelor în textul reconstituit. În cazul în care există mai multe interpretări ale frazei, va fi aleasă cea care conține cele mai puține cuvinte, deoarece se știe că strămoșii elfi erau foarte concisi.

Exemplu: *fraza.in* | *fraza.out*
amfostinvindecenarius | 5
12 | am
am | fost
fostin | invins
fost | de
decena | cenarius
orc
elf
narius
invins
de
cenarius
cena
rius

26. (Șirul naturii - *) Cu mii de ani înainte ca Fibonacci să descopere celebrul șir care îi poartă numele, elfii foloseau deja celebrele proprietăți ale acestui șir. După cum bine știți, elfii erau foarte apropiați de natură, așadar era destul de greu să nu observe că elementele acestui șir, precum și diverse numere derivate din acest șir, apar foarte des în natură. Sistemul Fibonacci a fost, la un moment dat, baza matematicii elfilor. Deși au descoperit destul de repede că bazele de numerație tradiționale ușurează calculele, micuții elfi erau nevoiți să învețe și sistemul de numerație pe care noi îl numim Fibonacci. Din motive evidente, acest sistem era numit de către elfi sistemul naturii, iar șirul era numit șirul naturii. Numerele scrise în sistemul naturii conțin numai cifre 0 și 1. Valoarea zecimală a unui număr scris în sistemul naturii este dată de formula $c_1 \cdot a_1 + c_2 \cdot a_2 + \dots + c_n \cdot a_n$, unde c_1, c_2, \dots, c_n sunt cifrele numărului scris în sistemul naturii, citite de la dreapta spre stânga, iar a_1, a_2, \dots, a_n sunt elementele semnificative ale șirului naturii ($a_1 = 1, a_2 = 2, a_3 = 3, a_4 = 5, a_5 = 8, a_6 = 13$, etc.). Să considerăm numărul 1010100000100101 scris în sistemul naturii. Valoarea zecimală a acestui număr este:

$$1 \cdot a_1 + 0 \cdot a_2 + 1 \cdot a_3 + 0 \cdot a_4 + 0 \cdot a_5 + 1 \cdot a_6 + 0 \cdot a_7 + 0 \cdot a_8 + 0 \cdot a_9 + 0 \cdot a_{10} + 0 \cdot a_{11} + 1 \cdot a_{12} + 0 \cdot a_{13} + 1 \cdot a_{14} + 0 \cdot a_{15} + 1 \cdot a_{16} = \\ 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 + 0 \cdot 5 + 0 \cdot 8 + 1 \cdot 13 + 0 \cdot 21 + 0 \cdot 34 + 0 \cdot 55 + 0 \cdot 89 + 0 \cdot 144 + 1 \cdot 233 + 0 \cdot 377 + 1 \cdot 610 + 0 \cdot 987 + 1 \cdot 1597 = \\ 1 + 3 + 13 + 233 + 610 + 1597 = 2457$$

Fișierul de intrare *sir.in* conține un singur număr scris în sistemul naturii. Numărul scris în sistemul naturii va conține cel mult 20 de cifre și este întotdeauna valid.

Fișierul de ieșire *sir.out* trebuie să conțină o singură linie care va avea numărul convertit în sistemul zecimal.

Exemplu:

<i>sir.in</i>	<i>sir.out</i>
1010100000100101	2457

27. (Multiplu - **) Să se determine, pentru un anumit număr n , dacă acesta poate avea un multiplu care să conțină doar o cifră dată x și cifra 0.

Fișierul de intrare *multiplu.in* conține pe prima linie numărul al cărui multiplu va fi determinat. Numărul al cărui multiplu este căutat este cuprins între 1 și 10000. Cea de-a doua linie a fișierului va conține cifra care poate apărea în multiplu, pe lângă cifra 0.

Fișierul de ieșire *multiplu.out* trebuie să conțină o singură linie, pe care se va afla multiplul determinat. Se garantează existența multiplului și faptul că acesta conține cel mult 10000 de cifre.

Exemplu:

<i>multiplu.in</i>	<i>multiplu.out</i>
7	1111110
1	

28. (Intervale - *)** Zăhărel are un interval $[1..L]$ ($L \leq 1.000.000.000$) și $N \leq 30.000$ alte intervale incluse în intervalul $[1..L]$. El poate atribui fiecăruia dintre cele N intervale o culoare, alb sau negru. Ajutați-l să determine o astfel de colorare cu proprietatea că, atât reuniunea intervalelor de culoare albă să fie intervalul $[1..L]$, cât și reuniunea intervalelor de culoare neagră să fie tot intervalul $[1..L]$.

Fișierul de intrare *int.in* conține, pe prima linie numerele L și N , cu semnificația din enunț. Pe următoarele N linii se găsesc perechi de numere întregi reprezentând intervalele.

Fișierul de ieșire *int.out* trebuie să conțină N linii; pe fiecare se va afla câte o cifră, 0 sau 1 (0 pentru negru și 1 pentru alb), linia i din fișierul de ieșire reprezentând culoarea intervalului de pe linia $i+1$ din fișierul de intrare. Dacă nu este posibilă o astfel de colorare se va afișa doar mesajul "Nu are soluție".

Exemplu

<i>int.in</i>	<i>int.out</i>
7 4	1
1 3	0
1 5	0
5 7	1
3 7	

29. (Palindrom prim - **) Un palindrom este un șir simetric, adică un șir care este la fel atât citit de la stânga la dreapta, cât și de la dreapta la stânga. Scrieți un program, care determină, pentru un interval $[A, B]$, toate palindroamele prime din interval.

În fișierul de intrare *palprim.in* vor fi, pe prima linie, numerele naturale A și $B \leq 100.000.000$.

În fișierul de ieșire *palprim.out* se vor scrie toate numerele palindroame și prime din intervalul dat, în ordine crescătoare.

Exemplu:

<i>palprim.in</i>	<i>palprim.out</i>
5 500	5
	7
	11
	101
	131
	151
	181
	191
	313
	353
	373
	383

30. (Lapte - **) Zăhărel are $N \leq 5.000$ vaci și, pentru, fiecare se știe intervalul de timp în care aceasta produce lapte. Determinați care este cel mai lung interval de timp în care, cel puțin o vacă produce lapte, și cel mai lung interval de timp în care nici o vacă nu produce lapte.

În fișierul de intrare *lapte.in* va fi, pe prima linie, numărul N . Pe următoarele N linii se găsesc perechi de numere întregi $\leq 1.000.000$, reprezentând intervalele.

În fișierul de ieșire *lapte.out* se vor scrie două numere reprezentând lungimea celui mai lung interval de timp în care, cel puțin o vacă produce lapte și lungimea celui mai lung interval de timp în care, nici o vacă nu produce lapte.

Exemplu:

3	lapte.in	900 300	lapte.out
300 1000			
700 1200			
1500 2100			

31. (Rebus - **) Se consideră un rebus tipic de N linii și M coloane ($N, M \leq 16.000$), care conține pătrățele albe și negre. Un cuvânt este o secvență continuă de lungime cel puțin 2, de pătrățele albe pe verticală sau orizontală. Să se determine, pentru un rebus dat, câte cuvinte pe verticală există și câte cuvinte pe orizontală.

În fișierul de intrare *rebus.in* vor fi, pe prima linie, numerele N și M . Pe următoarele N linii se găsesc numere naturale, primul reprezentând câte pătrățele negre există pe linia respectivă (linia i din fișier corespunde liniei $i-1$ din *rebus*), urmat de coloanele pe care se află pătrățelele negre. Se știe că nu vor exista mai mult de 30.000 de pătrățele negre.

În fișierul de ieșire *rebus.out* se vor scrie două numere reprezentând numărul de cuvinte verticale și numărul de cuvinte orizontale.

Exemplu:	rebus.in		rebus.out
5 5		4 5	
1 5			
2 3 4			
0			
2 2 3			
1 1			

32. **Editor - *****) Se consideră un editor de text care răspunde doar la apăsarea a șase taste, și anume cele care au tipărite simbolurile: "(", ")", "[", "]", "***" și "E". Dacă se apasă una din tastele "(", ")", "[", "]", atunci se afișează caracterul respectiv pe ecran. Dacă se apasă tasta "***" se șterge ultimul caracter afișat (dacă nu este afișat nici un caracter, atunci nu se întâmplă nimic). Dacă se apasă tasta "E" (Enter), atunci editorul va verifica dacă șirul afișat pe ecran este un șir parantezat corect. Un șir este parantezat corect dacă este construit conform regulilor:

$$\langle \text{sir parantezat corect} \rangle = \langle \text{sirul vid} \rangle:$$
$$\langle \text{şir parantezat corect} \rangle = "((" + \langle \text{şir parantezat corect} \rangle + "))"$$
$$\langle \text{șir parantezat corect} \rangle = "[" + \langle \text{șir parantezat corect} \rangle + "]" ;$$
$$\langle \text{sir parantezat corect} \rangle = \langle \text{sir parantezat corect 1} \rangle + \langle \text{sir parantezat corect 2} \rangle.$$

Prin $X + Y$ s-a notat concatenarea sirurilor X si Y .

De exemplu, șirurile "[([)])", "[([)])](O)" și "([([)])]" sunt parantezate corect, iar șirurile ")([)])", "[OOOO([([)])]" și "[([([)])]" nu sunt parantezate corect. Dându-se o succesiune de taste apășate care se termină cu tasta "E" și știind că, inițial, nu este nici un caracter afișat pe ecran, trebuie să decideți dacă șirul afișat pe ecran în urma apășării tastelor este un șir parantezat corect.

Pe prima linie a fișierului de intrare *editor.in* se află un număr întreg $T \leq 30$, reprezentând numărul de succesiuni de taste care vor fi descrise în continuare. Pe

fiecare din următoarele T linii este descrisă câte o succesiune de taste apăsat. Nici o succesiune de caractere nu va conține mai mult de 60000 de taste apăsat.

În fișierul *editor.out* vei scrie exact T linii, câte una pentru fiecare succesiune de taste descrisă în fișierul de intrare. Pe fiecare linie vei afișa ":" (fără ghilimele), dacă șirul obținut în urma succesiunii corespunzătoare de taste apăsată este parantezat corect, respectiv ":((fără ghilimele), în caz contrar.

<i>Exemplu:</i>	editor.in		editor.out
6	:)
E)	:)
))))*****]]]]*****E	:)
{([()[([]([]))][[]()([)](())])}([:)
][])E	:		(
[*]*[{}()*]()*E	:		(
({([)])E	:)
[E	:)

("Stelele informaticii" 2003)

33. (Templu - ***) Elfii doresc să construiască un mic templu de formă dreptunghiulară în mijlocul copacilor. Ei trebuie să descopere o zonă dreptunghiulară pe care să nu se afle nici un copac, deoarece ei nici măcar nu concep ideea de a tăia copacii. Pădurea poate fi privită ca o zonă dreptunghiulară, iar pozițiile copacilor sunt date prin coordonatele într-un sistem a cărui origine se află în colțul din stânga-jos a dreptunghiului. Dimensiunile copacilor sunt neglijabile, iar templul construit trebuie să ocupe o suprafață cât mai mare.

Prima linie a fișierului de intrare *templu.in* conține două numere separate printr-un spațiu, care reprezintă lungimea și lățimea pădurii. Cea de-a doua linie conține numărul n al copacilor din pădure. Numărul copacilor din pădure este cuprins între 1 și 200. Fiecare dintre următoarele n linii va descrie poziția unui copac. Această este reprezentată de coordonata orizontală și de coordonata verticală a copacului în sistemul de coordonate descris. Aceste două numere vor fi separate printr-un spațiu.

Fișierul de ieșire *templu.out* trebuie să conțină trei linii. Pe prima linie se va afla suprafața maximă care poate fi ocupată de templu. Pe cea de-a doua linie se vor afla două numere, separate printr-un spațiu, care reprezintă poziția colțului din dreapta-jos a templului. Pe cea de-a treia linie se vor afla două numere, separate printr-un spațiu, care reprezintă poziția colțului din stânga-sus a templului.

Exemplu:	templu.in	templu.out
10 10		50
3		10 0
3 5		0 5
3 8		
6 6		

34. (Traseu - ***) Alice și Bob vor să meargă în vacanță. Cei doi și-au planificat câte un traseu pe care doresc să-l urmeze și au scris o listă de orașe pe care vor să le viziteze într-o anumită ordine. O listă poate să conțină un anumit oraș de mai multe ori. Deoarece cei doi vor să călătorească împreună, trebuie să cadă de acord privind un traseu comun. Nici unul nu vrea să schimbe ordinea în care au planificat vizitarea orașelor conform listei proprii și nu vor să adauge orașe noi la ea. În concluzie, ei nu au altă posibilitate decât să ștergă anumite orașe din lista proprie. Bineînțeles, traseul comun trebuie să conțină cel mai mare număr de orașe posibile de vizitat. În regiune există exact 26 de orașe, care au fost codificate cu litere mici de la 'a' la 'z'.

Fișierul de intrare *traseu.in* conține două linii; prima linie conține lista lui Alice, cea de a doua, lista lui Bob. Fiecare listă conține cel puțin un caracter literă mică și cel mult 80 de caractere litere mici, care nu sunt separate prin nici un spațiu.

Fișierul de ieșire *traseu.out* va trebui să conțină toate traseele care respectă condițiile descrise mai sus și fiecare traseu se va scrie în fișier o singură dată. Pe o linie se va scrie un singur traseu. Se garantează existența a cel puțin unui traseu nevid și cel mult 1000 de trasee diferite.

<i>Exemplu:</i>	<i>traseu.in</i>		<i>traseu.out</i>
	ababca		ababa
	acbacba		abaca
			abcba
			acbca
			acaba
			acaca
			acbaa

35. (Paranteze - ***) Zăhărel se joacă adesea cu șiruri de paranteze (evident că folosește doar șiruri corecte!). Uneori ajunge să aibă secvențe mari de tipul ")))))).". Pentru a preveni astfel de situații a inventat "paranteza magică "]" care înlocuiește una sau mai multe paranteze ")", după cum este nevoie pentru a închide corect parantezele deschise. Determinați pentru un șir dat câte paranteze închise înlocuiește.

Fișierul de intrare *parantez.in* conține două linii: prima linie conține lungimea șirului, iar cea de a doua, șirul.

Fișierul de ieșire *parantez.out* conține atâtea linii câte paranteze magice sunt în șirul dat. Pe fiecare linie se va scrie câte paranteze închise înlocuiește fiecare paranteză magică, luate în ordinea în care apar în șirul dat de la stânga la dreapta.

<i>Exemplu:</i>	<i>parantez.in</i>		<i>parantez.out</i>
	8		3
	((((()))		1

CAPITOLUL 2

Subprograme Definite de Utilizator

2.1. Subprograme implementate în manieră iterativă

2.1.1 Teste cu alegere multiplă și duală

1. Ce se va afișa în urma executării programului următor:

<pre>var x,y,z:byte; procedure P(a,b:byte;var c:byte); var aux:byte; begin aux:=a; a:=b-a; c:=c-aux; b:=aux; end; begin x:=10; y:=100; z:=200; p(x,y,z); write(x,' ',y,' ',z,' '); p(x,y,z); write(x,' ',y,' ',z); end.</pre>	<pre>#include <iostream.h> int x,y,z; void P(int a,int b, int &c) { int aux; aux=a; a=b-a; c=c-aux; b=aux; } void main () { x=10; y=100; z=200; P(x,y,z); cout<<x<<" " <<y<<" " <<z<<" "; P(x,y,z); cout<<x<<" " <<y<<" " <<z<<" "; }</pre>
---	--

- | | |
|-------------------------|--------------------------|
| a) 90 10 190 176 90 100 | c) 10 100 190 10 100 190 |
| b) 10 10 190 10 10 180 | d) 10 100 190 10 100 180 |

2. Ce se va afișa în urma executării programului următor:

<pre>var a:byte; procedure P(x:byte; var y:byte); begin y:=y*x; x:=x+y; write(x,' ',y,' '); end; begin a:=11; p(a,a); writeln(a); end.</pre>	<pre>#include <iostream.h> int a =11; void P(int x,int &y) { y=y*x; x=x+y; cout<<x<<" " <<y<<" "; } void main(){ P(a,a); cout<<a; }</pre>
--	--

- | | |
|----------------|----------------|
| a) 11 121 11 | c) 242 242 242 |
| b) 132 121 121 | d) 11 121 121 |

3. În urma apelului $P(n, x)$ se dorește afișarea cifrei maxime a unui număr natural prin intermediul apelului $write(x)$, respectiv $cout << x$. Identificați antetul corect al subprogramului P .

- | | | | |
|----|-------------------------------------|----|------------------------|
| a) | function P(n:integer):byte; | a) | int P(int n) |
| b) | procedure P(var a,b:integer); | b) | void P(int &x, int &y) |
| c) | procedure P(n,x:integer); | c) | void P(int n, int x) |
| d) | procedure P(x:integer; var n:byte); | d) | void P(int x, int &n) |

4. Subprogramul P efectuează ștergerea dintr-un vector, transmis ca parametru, a tuturor elementelor egale cu o valoare cunoscută. Subprogramul primește, prin intermediul altor doi parametri întregi, lungimea tabloului și valoarea elementelor ce vor fi șterse. Identificați antetul corect al subprogramului P . Pentru varianta Pascal, se consideră următoarea definiție de tip: *type sir=array[1..100] of integer*.

- | | | | |
|----|---|----|-----------------------------------|
| a) | function P(a:sir;
n,x:integer):sir; | a) | int P(int a[100], int n, x) |
| b) | procedure P(var a:sir;
n,x: integer); | b) | void P(int a, int &x, int &y) |
| c) | procedure P(var a:sir; var n:byte;
x:integer); | c) | void P(int a[100], int &n, int x) |
| d) | procedure P(a:sir; n,x:integer); | d) | void P(int a[100], int x, int n) |

5. Ce valori vor fi afișate în urma executării următorului program?

```
var x:integer;

procedure p1(y:integer);
var x:integer;
begin
x:=y; x:=2; write(x, ' '); y:=x;
end;

procedure p2(var y:integer);
begin
x:=y; x:=3; write(x, ' '); y:=x;
end;
```

```
#include <iostream.h>
int x;
void p1(int y)
{ int x;
x=y; x=2;
cout<<x<<' '; y=x;
}

void p2(int &y)
{ x=y; x=3;
cout<<x<<' '; y=x;
}
```

```
begin
x:=1;
p1(x); write(x, ' ');
p2(x); writeln(x);
end.
```

- a) 2 2 3 3
b) 2 2 3 2

```
void main()
{ x=1;
p1(x); cout<<x<<' ';
p2(x); cout<<x<<endl;
}
```

- c) 2 1 3 3
d) 2 1 3 2

6. Ce valori vor fi afișate în urma executării următorului program?

```
var x:integer;

procedure p1(x:integer);
var y:integer;
begin
y:=10; x:=x + y; write(x, ' ');
end;

procedure p2(var x:integer);
var y:integer;
begin
y:=x; x:=x + 3;
x:=x + y; write(x, ' ');
end;

begin
x:=20;
p1(20); write(x, ' ');
p2(x); write(x, ' ');
end.
```

- a) 20 20 43 20
b) 30 20 43 43

```
#include <iostream.h>
int x;
void p1(int x)
{ int y=10;
x=x+y;
cout<<x<<' ';
}

void p2(int &x)
{int y;
y=x; x+=3; x+=y;
cout<<x<<' ';
}

void main()
{ x=20;
p1(20); cout<<x<<' ';
p2(x); cout<<x<<endl;
}
```

- c) 30 20 43 20
d) 20 20 46 46

7. Ce valori vor fi afișate în urma executării următorului program?

```
var x,y:integer;

procedure p(var a,b:integer);
var x:integer;
begin
x:=a*b; a:=a+x; b:=a+x;
write(a, ' ', b, ' ', x, ' ');
end;

begin
x:=5; y:=10;
p(x,y); write(x, ' ', y);
end.
```

- a) 55 105 50 105 105
b) 55 55 50 55 55

```
#include <iostream.h>
int x,y;

void p(int &a, int &b)
{ int x;
x=a*b; a+=x; b+=a+x;
cout<<a<<' '<<b<<' '<<x<<' ';
}

void main()
{ x=5; y=10;
p(x,y); cout<<x<<' '<<y;
```

- c) 55 105 50 55 55
d) 55 105 50 55 105

8. Ce valori vor fi afișate în urma executării următorului program?

```
var n:integer;
function cif(var x:integer):byte;
var c:byte;
begin
  c:=x mod 10; x:=x div 10;
  cif:=(c+x mod 10)mod 10;
end;
begin
  n:=21987;
  writeln(cif(n)+cif(n))
end.
```

```
#include <iostream.h>
int n=21987;
int cif(int &x)
{
  int c;
  c=x%10; x/=10;
  return (c+ x%10)%10;
}
void main()
{ cout << cif(n)+cif(n);
}
```

- a) 12; b) 10; c) 32; d) 30.

9. Indicați care dintre următoarele antete de funcții sunt corecte sintactic:

a) `function F1(z byte);`
 b) `function F2(x,y):real;`
 c) `function F3(y:byte):text;`
 d) `function 12F(c:char):real;`
 e) `function F4(y:byte;z:byte):real;`

a) `int F1(int z;)`
 b) `int F2(int x,y)`
 c) `FILE* F3(int y)`
 d) `float 12F(char c)`
 e) `float F4(int x, int y)`

10. Identificați care dintre funcțiile următoare returnează un număr real, reprezentând media aritmetică dintre câtul și restul la împărțirea a două numere întregi, transmise ca parametri:

a) `function M(x,y:integer):byte;
begin
 M := (x div y + x mod y)/2;
end;`
 b) `function M(x,y:integer):real;
var medie:real;
begin
 medie := x div y + x mod y;
 M := medie/2;
end;`
 c) `function M(x,y:integer):real;
begin
 M := x div y + x mod y;
 M := M/2;
end;`

a) `int M(int x,y)
{
 return (x/y + x*y)/2;
}`
 b) `float M(int x,int y)
{
 float medie;
 medie = x/y + x*y;
 return medie/2;
}`
 c) `float M(int x,int y)
{
 return x/y + x*y;
}`

d) `function M(x,y:integer):real;
begin
 x := x div y;
 y := x mod y;
 M := (x + y)/2;
end;`

d) `float M(int x,int y)
{
 x = x/y ;
 y = x*y ;
 return (x+y)/2;
}`

11. Considerăm definită funcția *prim*, care returnează *true* în varianta Pascal, respectiv 1 în varianta C++, dacă valoarea întreagă transmisă ca parametru este un număr prim și *false* / 0, în caz contrar. Considerăm un tablou unidimensional *a* ce conține *n* elemente de valori întregi. Identificați care dintre următoarele funcții returnează indicele din *a* al primului element număr prim. Dacă acesta nu conține nici un element prim, se returnează valoarea -1.

Pentru varianta Pascal, se consideră următoarea definiție de tip: *type sir=array[0..100] of integer*, deci primul element este *a[0]*.

a) `function p(a:sir;n:byte):integer;
var i:integer;
begin
 p:=-1;
 for i:=0 to n-1 do
 if prim(a[i]) then p:=i;
 end;`

a) `int p1(int a[101], int n)
{
 int i,x=-1;
 for(i=0;i<n;i++)
 if (prim(a[i])) x=i;
 return x;
}`

b) `function p(a:sir;n:byte):integer;
var x:integer;
begin
 x:=-1;
 repeat
 inc(x);
 until (prim(a[x])) or (x=n);
 if x=n then p:=-1
 else p:=x;
end;`

b) `int p2(int a[101], int n)
{
 int x=-1;
 do x++;
 while (!prim(a[x]) && (x<n));
 if (x==n) return -1;
 else return x;
}`

c) `function p(a:sir;n:byte):integer;
var x,i:integer;
begin
 x:=-1;
 while x<n-1 do begin
 inc(x);
 if prim(a[x])=true then x:=n;
 end;
 p:=x;
end;`

c) `int p3(int a[101], int n)
{
 int i,x=-1;
 while (x<n-1) {
 x++;
 if (prim(a[x])) return x;
 }
 return x-n-1;
}`

```

d)
function p(a:sir;n:byte):integer;
var i:integer;
begin
  p:=-1;
  for i:=0 to n-1 do
    if prim(i) then
      begin
        p:=1;
        exit;
      end
  end;
end;

```

12. Considerând următorul program, ce valori vor fi afișate în urma executării acestuia?

```

type sir=array[1..10] of integer;
var a:sir; i,n:integer;

procedure x(var a:sir;
            var n:integer);
var i,j:integer;
begin
  i:=1;
  while i<=n do
    if a[i]<0 then begin
      for j:=i to n-1 do
        a[j]:=a[j+1];
      dec(n);
    end
    else inc(i);
  end;

begin
  n:=6;
  a[1]:=-3;
  a[2]:=2; a[3]:=-1;
  for i:=1 to n div 2 do
    a[n-i+1]:=a[i];
  x(a,n);
  for i:=1 to n do write(a[i], ' ');
end.

```

- a) -3 -1 -1 -3
b) -2 -2

```

d)
int p4(int a[101], int n)
{
  int i,x=0;
  for(i=0;i<n;i++)
    if (prim(i)) x=i;
  return x;
}

```

```

#include<iostream.h>
int i,n=6,a[10];

void x(int a[10], int& n)
{
  int i=0,j;
  while (i<n) {
    if (a[i]<0){
      for(j=i;j<n-1;j++)
        a[j]=a[j+1];
      n--;
    }
    else i++;
  }
}

void main()
{
  a[0]=-3;
  a[1]=2;
  a[2]=-1;
  for(i=n/2;i<n;i++)
    a[i]=a[n-i-1];
  x(a,n);
  for(i=0;i<n;i++)
    cout<<a[i]<<' ';
}

```

- c) 2 2
d) 3 3 3 3

13. Considerăm un tablou unidimensional a , ce conține n elemente de valori întregi. Identificați care dintre următoarele funcții returnează indicele din a al primului element de valoare 0. Dacă acesta nu conține nici un element nul, se returnează valoarea -1. Pentru varianta Pascal, se consideră următoarea definiție de tip: `type sir=array[0..100] of integer`, deci primul element este $a[0]$.

```

a)
function p(a:sir;n:byte):integer;
var i:integer;
begin
  p:=-1;
  for i:=0 to n-1 do
    if a[i]=0 then p:=i;
  end;
end;

```

```

b)
function p(a:sir;n:byte):integer;
var x:integer;
begin
  x:=-1;
  repeat
    inc(x)
  until (a[x]=0) or (x=n);
  p:=x; end;

```

```

c)
function p(a:sir;n:byte):integer;
var x,i:integer; ok:boolean;
begin
  x:=-1; ok:=true;
  while (x<n-1) and ok do begin
    inc(x);
    if (a[x]=0) then ok:=false;
  end;
  if ok then p:=-1 else p:=x;
end;

```

```

d)
function p(a:sir;n:byte):integer;
var x:integer;
begin
  x:=-1;
  while ((a[x+1]<>0) and (x<n-1)) do
    inc(x);
  p:=x+1;
end;

```

14. Ce se va afișa în urma executării următorului program:

```

var a:string; var b:char;

procedure x(var s:string;
            var c:char);
begin
  while c=s[length(s)] do
    delete(s,length(s),1);
  if length(s)>0 then
    c:=uppercase(s[length(s)])
  else c:='.';
end;

```

```

a)
int p(int a[101], int n)
{
  int i,x=0;
  for(i=0;i<n;i++)
    if (a[i]==0) x=i;
  return x;
}

```

```

b)
int p(int a[101], int n)
{
  int x=-1;
  do x++;
  while ((a[x]!=0) && (x<n));
  return x;
}

```

```

c)
int p(int a[101], int n)
{
  int i,x=-1;
  while (x<n-1)
  {
    x++;
    if (a[x]==0)
      return x;
  }
  return x-n;
}

```

```

d)
int p(int a[101], int n)
{
  int x=-1;
  while ((a[x+1]!=0) && (x<n))
    x++;
  return x+1;
}

```

```

#include <iostream.h>
#include <string.h>
char *a, b;

void x(char *s, char &c) {
  while (c==s[strlen(s)-1])
    s[strlen(s)-1]=0;
  if (strlen(s)>0)
    c=s[strlen(s)-1],
    c<='a' && c<='z'?c+'A':-'a':c;
  else c='.';
}

```



```
begin
  a:='copiii'; b:='i';
  x(a,b);
  writeln(a,b);
end.
```

- a) copP b) copiii. c) copiiii d) copiiiP

15. Care dintre următoarele funcții returnează cea mai mare putere a lui 2 care este mai mică sau egală cu o valoare naturală (<15), transmisă ca parametru?

```
a)
function F(n:byte):integer;
var x:integer;
begin
  x:=-1;
  repeat
    inc(x);
  until 1 shl x>n;
  F:=1 shl (x-1);
end;
```

```
b)
function F(n:byte):integer;
var x:integer;
begin
  x:=1;
  while (x<n) do x:=x*2;
  F:=x;
end;
```

```
c)
function F(n:byte):integer;
var x:integer;
begin
  x:=1;
  repeat
    x:=x*2;
  until x>n;
  F:=x-1;
end;
```

```
d)
function F(n:byte):integer;
var i,x:integer;
begin
  x:=1;
  for i:=1 to n do begin
    x:=x*2;
    if x<=n then begin
      F:=x; exit;
    end;end;
  F:=x-1;
end;
```

```
void main() {
  a = "copiii"; b = 'i';
  x(a,b);
  cout<<a<<b<<endl;
}
```

```
a)
int F(int n)
{
  int x=-1;
  do
    x++;
  while (1 << x <= n );
  return 1 << (x-1);
}
```

```
b)
int F(int n)
{
  int x=1;
  while (x<n) x*=2;
  return x;
}
```

```
c)
int F(int n)
{
  int x=1;
  do
    x*=2;
  while (x<n);
  return x-1;
}
```

```
d)
int F(int n)
{
  int i,x;
  x=1;
  for(i=1;i<=n;i++)
  {
    x*=2;
    if (x<=n) return x;
  }
  return x-1;
}
```

16. Ce se va afișa în urma executării următorului program:

```
var a:real;
function F(var x:real):real;
begin
  x:=abs(x*10);
  F:=int(x/10) + frac(x);
end;
begin
  a:=15.25;
  write(F(a):0:2, ' ');
  write(F(a):0:2, ' ');
  writeln(a:0:2);
end.
```

- a) 15.50 15.50 1525.00
b) 15.50 15.50 152.50

```
#include <stdio.h>
float F(float &x)
{ x=(x > 0 ? x: -(x));
  x*=10;
  return int(x/10) + x - int(x);
}
void main()
{float a=15.25;
 printf("%0.2f ",F(a));
 printf("%0.2f ",F(a));
 printf("%0.2f ",a);
}
```

- c) 15.50 152.00 1525.00
d) 15.50 152.00 152.50

17. Câte elemente divizibile cu 10 se vor afișa în urma executării programului următor:

```
var x,y:byte;
function F(var y:byte;x:byte):byte;
begin
  y:=y div 10 + x;
  F:= x+y ;
end;
begin
  x:=101; y:=10;
  write(F(x,y), ' ');
  write(x, ' ', y, ' ');
  writeln(F(x,y));
end.
```

- a) 1 b) 2 c) 3 d) 4

```
#include <iostream.h>
int F(int &y, int x)
{
  y = y/10 + x;
  return x + y;
}
main()
{int x = 101, y = 10;
 cout<<F(x,y)<<" ";
 cout << x <<" "<< y<<" ";
 cout<<F(x,y);
}
```

18. În urma apelului $write(P(n, x))$, respectiv $cout<<P(n, x)$, se dorește afișarea numărului de apariții al cifrei x în scrierea zecimală a numărului întreg n . Identificați antetul corect al subprogramului P .

```
a)
procedure P(n,x:integer)

b)
function P(x,c:integer):integer

c)
function P(n:real;x:byte):byte

d)
function P(x:integer;n:char):byte
```

```
a)
void P(int n, int x)

b)
int P(int x, int c)

c)
int P(int n[10], int x)

d)
void P(int x, float n)
```

19. Considerăm subprogramul următor:

```
function P(var x:integer):byte;
var s:byte;
begin
  s:=0;
  while x>0 do begin
    s:=s+x mod 2;  x:=x div 2;
  end;
  P:=s;
end;
```

```
int P(int &x)
{ int s=0;
  while (x!=0)
  {
    s+=x%2;
    x/=2;
  }
  return s;
}
```

Considerăm că variabila întregă n , declarată global, are valoarea 32. Ce se va afișa în urma apelului $write(P(n)+P(n)+1)$, respectiv $cout<<P(n)+P(n)+1$?

- a) 2; b) 3; c) 4; d) 1.

20. Considerăm secvența de declarații următoare:

```
var x:array[1..10] of byte;

procedure P1(n:integer);
var x:integer;
begin
  ...
end;

procedure P2(n,x:real);
begin
  ...
end;

begin
  ...
end.
```

```
int x[100];

void P1(int n)
{ char x;
  ...
}

void P2(float n,float x)
{
  ...
}

void main ()
{
  ...
}
```

Care dintre următoarele afirmații este adevărată:

- a) Elementul $x[1]$ poate fi referit oriunde în program, deci în oricare subprogram $P1$ sau $P2$;
 b) Programul sursă conține erori de sintaxă;
 c) Elementul $x[1]$ poate fi referit doar în programul principal (varianta Pascal), respectiv numai în funcția $main$ (varianta C++);
 d) Elementul $x[1]$ poate fi referit doar în subprogramele $P1$ și $P2$.

2.1.2 Probleme rezolvate

1. Se consideră un vector cu n ($1 \leq n \leq 40$) componente numere naturale mai mici sau egale cu 60000. Să se realizeze un subprogram care, primind prin intermediul a doi parametri tabloul respectiv și lungimea sa, întoarce printr-un alt parametru, cel mai mare element prim. Dacă nu există, se va returna valoarea -1. Considerăm că există definită funcția *prim* care verifică dacă valoarea naturală primită ca parametru este un număr prim. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu număr prim.

Exemplu: Pentru $n=5$ și tabloul (1, 6, 7, 55, 19), subprogramul va returna prin intermediul unui parametru valoarea "19".

Soluție:

Subprogramul va avea trei parametri, dintre care unul referință. Datorită faptului că valoarea întoarsă poate fi -1, tipul parametrului nu poate aparține un tip de date fără semn. Pentru varianta Pascal este definit tipul de date *sir*, iar subprogramul realizat este o procedură, datorită faptului că rezultatul este returnat prin intermediul unui parametru. Tot din acest motiv, în varianta C++, funcția realizată are tipul *void*.

```
1 type sir=array[1..40] of word;
2 procedure P(x:sir; n:byte;
3             var y:longint);
4   var i:byte;
5   begin
6     y:=-1;
7     for i:=1 to n do
8       if (prim(x[i]))and(y<x[i])
9       then y:=x[i];
10    end;
```

```
void P(unsigned int x[40],
        int n, long &y)
{
  int i;
  y = -1;
  for(i = 0; i < n; i++)
    if (prim(x[i])&&(x[i]>y))
      y = x[i];
}
```

2. Se consideră un vector cu n ($1 \leq n \leq 50$) componente numere naturale, mai mici sau egale cu 30000. Să se realizeze un subprogram care, primind prin intermediul a doi parametri tabloul respectiv și lungimea sa, afișează toate elementele care conțin un număr maxim de cifre în reprezentarea în baza 10. Considerăm că există definită funcția *nrc* care returnează, pentru o valoare naturală primită printr-un parametru, numărul de cifre din reprezentarea acesteia în baza 10.

Exemplu: Pentru $n=4$ și tabloul (231, 54, 809, 2), subprogramul va afișa 231, 809.

Soluție:

Subprogramul va avea doi parametri valoare. Pentru varianta Pascal, subprogramul realizat este o procedură, datorită faptului că operația pe care trebuie să o realizeze este cea de afișare, fără să fie necesar transferul unor rezultate. Tot din acest motiv, în varianta C++, funcția realizată are tipul *void*. Pentru varianta Pascal se consideră definit tipul de date *sir=array[1..50] of word*;

```

1 procedure P(x:sir; n:byte);
2 var m,i:byte;
3 begin
4 m:=0;
5 for i:=1 to n do
6 if (nrc(x[i])>m) then
7 m:=nrc(x[i]);
8 for i:=1 to n do
9 if (nrc(x[i])=m) then
10 write(x[i],' ');
11 end;

void P(int x[50],int n)
{
int i,m=0;
for (i=0; i<n;i++)
if (nrc(x[i])>m)
m=nrc(x[i]);
for (i=0; i<n;i++)
if (nrc(x[i])==m)
cout<<x[i]<<' ';
}

```

3. Se citesc de la tastatură n ($1 \leq n \leq 50$) intervale de numere întregi de forma $[a, b]$, introduse prin limitele a și b ($-100 \leq a \leq b \leq 100$). Să se realizeze un subprogram care efectuează operația de citire a intervalelor și returnează numărul maxim de valori palindrom incluse într-unul dintre intervalele citite, fără a folosi vreun parametru. Considerăm că există definită funcția *pal* care verifică dacă valoarea naturală primită printr-un parametru reprezintă un număr palindrom. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu palindrom.

Exemplu: Pentru $n=4$ și intervalele $[32, 44]$, $[2, 4]$, $[5, 12]$, $[56, 62]$, subprogramul va returna valoarea 6, deoarece intervalul $[5, 12]$ conține 6 numere palindrom.

Soluție:

În enunț se specifică faptul că rezultatul nu va fi transferat prin intermediul parametrilor referință. Deoarece tipul valorii rezultat aparține unui tip de date simplu, este posibilă realizarea unei funcții al cărei rezultat va fi de tip *byte* / *unsigned int*.

Funcția nu va realiza transfer de date prin parametri valoare, deoarece limitele intervalelor citite și numărul acestora vor reprezenta variabile locale.

```

1 function citește:integer;
2 var i,j,nr,m,n,a,b:byte;
3 begin
4 m:=0; readln(n);
5 for i:=1 to n do begin
6 readln(a,b); nr:=0;
7 for j:=a to b do
8 if pal(j) then inc(nr);
9 if nr>m then m:=nr;
10 end;
11 citește:=m;
12 end;

unsigned int citește()
{
int i,j,nr,m,n,a,b;
m=0; cin>>n;
for (i=1;i<=n;i++){
cin>>a; cin>>b; nr=0;
for (j=a;j<=b;j++)
if (pal(j)) nr++;
if (nr>m) m=nr;
}
return m;
}

```

4. Subprogramul *L* permite înlocuirea elementelor unui vector *a*, transmis ca parametru, cu suma factorialilor cifrelor lor, dacă această sumă nu se află într-un interval $[x,y]$. Subprogramul va primi prin intermediul a trei parametri întregi, lungimea vectorului n ($n \leq 50$) și limitele intervalului. El va face apel la funcția *S*,

care returnează suma factorialilor cifrelor unui număr natural transmis printr-un parametru. Scrieți definițiile complete ale subprogramelor *L* și *S*. Inițial, elementele vectorului sunt numere întregi mai mici ca 10.000.

Exemplu: Pentru $n=5$, $x=5$ $y=35$ și tabloul $a=(10, 234, 21, 148, 34)$ la finalul execuției subprogramului *L*, elementele lui *a* vor fi $(2, 234, 3, 40345, 34)$.

Soluție:

Subprogramul va avea patru parametri, dintre care unul referință (tabloul). Pentru varianta Pascal este definit tipul de date *sir*, iar subprogramul realizat este o procedură, datorită faptului că rezultatele sunt returnate prin intermediul unui parametru (ce nu aparține unui tip de date simplu). Tot din acest motiv, în varianta C++, funcția realizată are tipul *void*.

Valoarea returnată de funcția *S* este preluată în variabila locală *v*, pentru a se evita apelul repetat pentru o singură valoare a parametrului.

```

1 type
2 sir=array[1..50]of longint;
3
4 function S(x:integer):longint;
5 var sm,p,i:longint;
6 begin
7 sm:=0;
8 while x<>0 do begin
9 p:=1;
10 for i:=1 to x mod 10 do
11 p:=p*i;
12 sm:=sm+p; x:=x div 10;
13 end;
14 s:=sm;
15 end;

long S (int x)
{
long s,p,i;
s=0;
while (x!=0) {
p=1;
for (i=1;i<=x%10;i++)
p*=i;
s+=p;
x/=10;
}
return s;
}

16 procedure L(var a:sir;
17 n,x,y:integer);
18 var i,v:longint;
19 begin
20 for i:=1 to n do begin
21 v:=S(a[i]);
22 if (v<x) or (v>y) then a[i]:=v;
23 end;
24 end;

void L(long a[50],int n,
int x,int y)
{
long i,v;
for (i=0;i<n;i++){
v=S(a[i]);
if ((v<x) || (v>y)) a[i]=v;
}
}

```

5. Realizați un subprogram *S*, care să permită ordonarea elementelor unui vector *a* de numere întregi, transmis ca parametru. Subprogramul va primi, prin intermediul parametrilor întregi *x* și *y*, indicii elementelor între care se va face sortarea. Tipul ordonării care se dorește a fi realizată (ascendentă-descendentă) va fi transmis codificat prin parametrul întreg *k*. Dacă la apel, acestuia *i* se transmite valoarea 1, atunci se va realiza o ordonare ascendentă, iar dacă valoarea transmisă acestuia este -1, sortarea va fi descendentă.

Exemplu: Considerăm vectorul $a=(10, 234, 21, 1, 34)$. Subprogramul *S* apelat de două ori, pentru ordonarea crescătoare a primelor trei elemente și descrescătoare a următoarelor două, modifică elementele tabloului astfel: $(10, 21, 234, 34, 1)$.

Soluție:

Subprogramul va avea patru parametri, dintre care unul referință (tabloul a). Pentru varianta Pascal este definit tipul de date $sir=array[1..50]$ of integer, iar subprogramul realizat este o procedură, datorită faptului că rezultatele sunt returnate prin intermediul unui parametru (ce nu aparține unui tip de date simplu). Tot din acest motiv, în varianta C++ funcția realizată are tipul *void*. Inegalitatea care intervine în procesul de ordonare $k*a[i]>k*a[j]$ își schimbă semnul când valoarea transmisă lui k este -1, realizându-se astfel o ordonare descrescătoare.

<pre>1 procedure S(var a:sir; 2 k,x,y:integer); 3 var i,j,t:integer; 4 begin 5 for i:=x to y-1 do 6 for j:=i+1 to y do 7 if k*a[i]>k*a[j] then 8 begin 9 t:=a[i]; 10 a[i]:=a[j]; 11 a[j]:=t; 12 end; 13 end;</pre>	<pre>void S(int a[50],int k, int x,int y) { int i,j,t; for (i=x;i<y;i++) for (j=i+1;j<=y;j++) if (k*a[i]>k*a[j]){ t=a[i]; a[i]=a[j]; a[j]=t; } }</pre>
--	---

6. Funcția S verifică dacă un tablou unidimensional, primit prin intermediul parametrului a , reprezintă o permutare, fără puncte fixe, a mulțimii numerelor de la 1 la n ($n \leq 50$, transmis ca parametru întreg). Subprogramul returnează, în Pascal, valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum tabloul primit reprezintă sau nu o permutare fără puncte fixe.

Spunem că o permutare (a_1, a_2, \dots, a_n) are puncte fixe dacă există cel puțin un element $a_i = i$ ($1 \leq i \leq n$). De exemplu, pentru $n=4$, permutarea (2,4,3,1) are puncte fixe deoarece $a_3=3$.

Subprogramul S va apela funcția nr , care determină numărul de apariții al unei valori într-un vector. Atât valoarea căutată, cât și tabloul și lungimea acestuia sunt primite de către funcția nr prin intermediul parametrilor.

Scrieți definițiile complete ale subprogramelelor S și nr .

Exemplu: Pentru $n=5$ și oricare dintre tablourile (2, 4, 5, 1, 2) sau (2, 1, 5, 4, 3) se va returna valoarea *False*/0.

Soluție:

Funcția S va avea doi parametri: tabloul a și n , lungimea acestuia. Funcția nr va fi apelată pentru a verifica dacă tabloul a reprezintă o permutare a mulțimii numerelor de la 1 la n . Pentru varianta Pascal este definit tipul de date $sir=array[1..50]$ of byte.

<pre>1 function nr(a:sir; 2 n,x: byte):byte; 3 var m:byte; 4 begin 5 m:=0; 6 for i:=1 to n do 7 if a[i]=x then inc(m); 8 nr:=m; 9 end;</pre>	<pre>int nr(int a[50],int n,int x) { int m=0; for (i=0;i<n;i++) if (a[i]==x) m++; return m; }</pre>
<pre>10 11 function S(a:sir;n:byte): 12 boolean; 13 var i:byte; 14 begin 15 S:=true; 16 for i:=1 to n do 17 if (nr(A,n,i)<>1)or(a[i]=i) 18 then S:=false; 19 end;</pre>	<pre>int S(int a[50],int n) { int i; for (i=1;i<=n;i++) if ((nr(a,n,i)!=1) (a[i-1]==i)) return 0; return 1; }</pre>

7. Subprogramul X primește, prin intermediul a doi parametri, un tablou unidimensional de numere întregi și numărul de elemente al acestuia (≤ 100). El returnează, prin intermediul altor doi parametri întregi, cifra maximă ce apare în scrierea din baza 10 a elementelor vectorului și numărul de apariții al acesteia. În cadrul subprogramului X se apelează subprogramul D , care determină, pentru o valoare întreagă, transmisă ca parametru, cifra maximă a sa și numărul de apariții al acesteia. Aceste două valori sunt returnate de D prin intermediul a doi parametri întregi.

Scrieți definițiile complete ale celor două subprograme.

Exemplu: Pentru tabloul ce conține 5 elemente: (22, 405, 5125, 102, 53), subprogramul X va returna două valori 5 și 4, reprezentând cifra maximă a elementelor din vector și numărul de apariții al acesteia.

Soluție:

Pentru varianta Pascal, ambele subprograme vor fi implementate ca proceduri, rezultatele fiind returnate, pentru fiecare, prin intermediul a doi parametri referință. Pentru varianta C++, funcțiile vor fi, din același motiv, de tip *void*. Subprogramul D va determina, pentru fiecare element din vector, cifra maximă și numărul de apariții al acesteia.

Pentru varianta Pascal considerăm definit tipul de date $sir=array[1..100]$ of integer.

<pre>1 procedure D(x:integer; 2 var c,m:byte); 3 begin 4 m:=0; c:=0; 5 while x<>0 do begin 6 if c<x mod 10 then begin 7 c:=x mod 10; m:=1; 8 end</pre>	<pre>void D(int x,int &c, int &m) { m=0; c=0; while (x!=0){ if (c<x%10){ c=x%10; m=1; } }</pre>
---	--

```

9   else
10    if c=x mod 10 then inc(m);
11    x:=x div 10;
12  end;
13 end;
14
15 procedure X(a:sir; n:byte;
16             var cm,nm:byte);
17 var i,c,m:byte;
18 begin
19   cm := 0; nm := 0;
20   for i:=1 to n do begin
21     D(a[i],c,m);
22     if c=cm then nm:=nm+m
23     else if c>cm then begin
24       cm:=c;
25       nm:=m
26     end
27   end
28 end;
29
30 else
31   if (c==x % 10) m++;
32   x=x / 10;
33 }
34
35 void X(int a[100],int n,
36       int &cm,int &nm)
37 {
38   int i,c,m;
39   cm = 0; nm = 0;
40   for (i=0;i<n;i++){
41     D(a[i],c,m);
42     if (c==cm) nm+=m;
43     else
44       if (c>cm)
45         {cm=c; nm=m;}
46   }
47 }

```

8. Considerăm o matrice pătratică cu n linii și n coloane ($n \leq 10$), cu elemente valori strict pozitive. Funcția L determină numărul de ordine al liniei ce conține cele mai multe elemente care sunt termeni ai șirului lui Fibonacci. Matricea și numărul n de linii sunt transmise funcției prin intermediul a doi parametri.

În cadrul funcției L se va face apel la funcția Fib care verifică dacă o valoare naturală transmisă ca parametru reprezintă un termen al șirului lui Fibonacci. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu termen al șirului.

Scrieți definițiile complete ale subprogramelor L și Fib .

Exemplu: Pentru $n=3$ și tabloul :

```

2 14 8
3 5 21
21 4 4

```

funcția L va returna valoarea 2.

Soluție:

Funcția Fib verifică dacă o valoare naturală x reprezintă un termen al șirului lui Fibonacci folosind trei variabile locale de tip întreg. Ea este apelată de funcția L pentru fiecare element din matrice, contorizându-se în același timp numărul de valori corecte de pe fiecare linie. Pentru varianta Pascal, considerăm definit tipul de date $mat=array[1..10,1..10]$ of integer.

```

1 function Fib(x:integer):boolean;
2 var a,b,c:integer;
3 begin
4   a := 1; b := 1;
5   while a+b<=x do begin
6     c := a + b; a := b; b := c;
7   end;
8   fib := (c = x) or (x = 1);
9 end;
10
11 int Fib(int x)
12 {
13   int a,b,c;
14   a = 1; b = 1;
15   while (a + b<=x){
16     c = a + b; a = b; b = c;
17   }
18   return (c==x) || (x==1);
19 }

```

```

10 Function L(a:mat; n:byte):byte;
11 var i,j,m,mx:byte;
12 begin
13   mx := 0;
14   for i:=1 to n do begin
15     m := 0;
16     for j:=1 to n do
17       if Fib(a[i,j]) then inc(m);
18     if m > mx then begin
19       L := i; mx := m;
20     end
21   end;
22 end;
23
24 int L(int a[10][10],int n)
25 {int l,i,j,m,mx;
26   mx=0;
27   for (i=0;i<n;i++){
28     m=0;
29     for (j=0;j<n;j++){
30       if (Fib(a[i][j])) m++;
31       if (m > mx){
32         l = i; mx = m;
33       }
34     }
35   }
36   return l+1;
37 }

```

9. Fișierul text *in.txt* conține, pe o singură linie, mai multe numere naturale mai mici ca 30000, separate prin câte un spațiu. Funcția Z permite citirea tuturor valorilor din fișier și returnează numărul de zerouri în care se termină produsul valorilor citite.

În cadrul ei este apelat subprogramul Exp care returnează, prin intermediul unui parametru întreg, exponentul la care apare un număr prim în descompunerea unui număr natural. Ambele valori sunt primite de subprogram prin intermediul a doi parametri întregi.

Scrieți definițiile complete ale subprogramelor Z și Exp .

Exemplu: Considerând că fișierul *in.txt* conține valorile 24, 25, 15, 10, 150, atunci funcția Z va returna valoarea 5.

Soluție:

Numărul de zerouri în care se termină produsul unor numere este egal cu cel mai mic exponent al factorilor de 2 și de 5, care apar în descompunerea lor.

Funcția Z va apela subprogramul Exp pentru a determina acești exponenți. În varianta Pascal, subprogramul Exp va fi implementat ca procedură deoarece se impune ca valoarea exponentului să fie returnată lui X prin intermediul unui parametru referință. Din același motiv, în varianta C++, funcția Exp va avea tipul *void*.

```

1 procedure Exp(x,f:integer;
2             var e:integer);
3 begin
4   while x mod f=0 do begin
5     inc(e);
6     x:=x div f;
7   end;
8 end;
9
10 function Z:integer;
11 var f:text;e5,e2,x:integer;
12 begin
13   e5:=0; e2:=0;
14   assign(f,'in.txt'); reset(f);
15
16 void Exp(int x,int f,int &e)
17 {
18   while (x%f==0)
19   {
20     e++;
21     x/=f;
22   }
23 }
24
25 int Z()
26 {
27   FILE *f; int e5,e2,x;
28   e5=e2=0;
29   f=fopen("in.txt","r");

```

```

15 while not eof(f) do begin
16   read(f,x);
17   Exp(x,5,e5); Exp(x,2,e2);
18 end;
19 close(f);
20 if e5<e2 then Z:=e5
21 else Z:=e2;
22 end;

```

```

while (fscanf(f,"%d",&x)==1)
{
  Exp(x,5,e5); Exp(x,2,e2);
}
fclose(f);
return e5<e2 ? e5 : e2;
}

```

10. Se consideră un număr natural x , citit de la tastatură. Să se realizeze un program care determină cel mai apropiat număr de x care reprezintă factorialul unei valori. Programul va conține două subprograme:

- funcția P , care verifică dacă o valoare transmisă ca parametru poate fi scris sub forma $k!$ ($1 \leq k$)
- funcție A , care are doi parametri întregi x și t . Ea determină cel mai mic număr mai mare ca x sau cel mai mare număr mai mic decât x de forma $k!$, după cum valoare unui parametru t este 1 sau -1 la apel.

Exemplu: Pentru $x=10$ se va afișa 6 ($6=3!$) iar pentru $x=22$ se va afișa 24 ($24=4!$).

Soluție:

Funcția A va mări sau micșora valoarea inițială a parametrului x , până când acesta va avea o valoare de forma $k!$. Valoarea primită la apel de parametru valoare t va reprezenta valoarea cu care se va modifica x în cadrul fiecărei iterații.

Funcția P returnează, în Pascal, valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului reprezintă sau nu factorialul unei valori.

```

1 var x:integer;
2 function P(x:integer):boolean;
3 var i,y:integer;
4 begin
5   y:=1; i:=1;
6   while y<x do begin
7     y:=y*i; inc(i)
8   end;
9   p:= y=x;
10 end;
11
12 function A(x,t:integer):integer;
13 begin
14   while (not P(x)and(x>0)) do
15     x:=x+t;
16   A:=x;
17 end;
18
19 begin
20   readln(x);
21   if x-A(x,-1)<A(x,1)-x then
22     write(A(x,-1))
23   else write(A(x,1));
24 end.

```

```

#include <iostream.h>
int x;

int P(int x)
{ int i,y;
  y=1; i=1;
  while (y<x) {
    y*=i; i++;
  }
  return (y==x);
}

int A(int x,int t)
{
  while (!P(x)&&(x>0)) x+=t;
  return x;
}

void main ()
{ cin>>x;
  if (x-A(x,-1)<A(x,1)-x)
    cout<<A(x,-1);
  else cout<<A(x,1);
}

```

11. Considerăm o matrice pătratică cu n linii și n coloane ($n \leq 10$), cu elemente valori strict pozitive. Funcția L determină numărul de ordine al liniei ce conține cele mai multe elemente care sunt valori autopomorfe. Numim număr autopomorfic o valoare care este egală cu unul dintre sufixele pătratului său. Exemplu de valoare autopomorfică este 25 deoarece $25^2 = 625$.

Matricea și numărul n de linii ale sale sunt transmise funcției prin intermediul a doi parametri.

În cadrul funcției L se va face apel la funcția A , care verifică dacă o valoare naturală transmisă ca parametru este autopomorfică. Ea returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++, după cum valoarea parametrului este sau nu număr autopomorfic.

Scrieți definițiile complete ale subprogramelor L și A .

Exemplu: Pentru $n=3$ și tabloul :

```

2 5 8
3 5 25
6 4 4

```

funcția L va returna valoarea 2.

Soluție

Funcția A verifică dacă o valoare naturală x este număr autopomorfic folosind o singură variabilă locală care este egală inițial cu pătratul lui x .

Ea este apelată de funcția L pentru fiecare element din matrice, contorizându-se în același timp numărul de valori corecte de pe fiecare linie.

Pentru varianta Pascal, considerăm definit tipul de date *mat=array[1..10,1..10] of integer*.

```

1 function A(x:integer):boolean;
2 var y:integer;
3 begin
4   y := x * x; A:=true;
5   while x>0 do begin
6     if y mod 10<>x mod 10 then
7       begin
8         A:=false; exit;
9       end;
10    x:=x div 10; y:=y div 10;
11  end; end;
12
13 function L(b:mat; n:byte):byte;
14 var i,j,m,mx:byte;
15 begin
16   mx := 0;
17   for i:=1 to n do begin
18     m := 0;
19     for j:=1 to n do
20       if A(b[i,j]) then inc(m);
21     if m > mx then begin
22       L := i; mx := m;
23     end;
24 end;

```

```

int A(int x)
{ int y;
  y = x * x;
  while (x>0){
    if (y % 10!=x % 10)
      return 0;
    x/=10;
    y/=10;
  }
  return 1;
}

int L(int b[10][10],int n)
{ int i,j,m,mx;
  mx = 0;
  for (i=0;i<n;i++){
    m = 0;
    for (j=0;j<n;j++){
      if (A(b[i][j])) m++;
      if (m > mx){
        i = i; mx = m;
      }
    }
  }
  return i+1;
}

```

12. Funcția S primește, prin intermediul parametrului a , un vector de numere întregi cu 10 de elemente și, prin intermediul parametrului k , un număr natural nenul ($k \leq 10$). Funcția returnează suma tuturor elementelor pozitive $a[i]$ cu proprietatea că $k \leq i \leq 10$. Realizați un program care citește de la intrarea standard un șir de 10 de valori întregi și două numere naturale nenule x și y ($x < y \leq 10$) și afișează suma elementelor pozitive din șir cu numerele de ordine cuprinse între x și y , folosind apeluri la funcția S .

Exemplu: Pentru șirul 2, 5, 3, -7, -8, -9, 4, 2, 3, 3, $x=3$ și $y=8$ programul va afișa valoarea 9. ($3+4+2=9$)

Soluție:

Pentru a determina suma elementelor pozitive din șir cu numerele de ordine cuprinse între x și y (inclusiv x și y), se va apela funcția S pentru $k = x$ și $k = y + 1$. Programul va afișa diferența celor două valori returnate de funcție. Ambii parametri ai funcției realizează un transfer prin valoare.

```
1  type                               #include <iostream.h>
2  sir= array[1..10] of integer;      int i,x,y, v[11];
3  var i,x,y:integer; v:sir;
4
5  function S(a:sir;k:byte):word;      int S(int a[11], int k)
6  var t,i:integer;                   {int t,i;
7  begin                               t=0;
8  t:=0;                               for (i=k;i<=10;i++)
9  for i:=k to 10 do                   if (a[i]>0) t+=a[i];
10 if a[i]>0 then t:=t+a[i];           return t;
11 S:=t;                               }
12 end;
13
14 begin
15 for i:=1 to 10 do read(v[i]);
16 readln(x,y);
17 writeln(S(v,x)-S(v,y+1));
18 end;

void main()
{
for (i=1;i<=10;i++)
cin>>v[i];
cin>>x>>y;
cout<<(S(v,x)-S(v,y+1));
}
```

13. Realizați un program care să verifice dacă o valoare naturală x ($2 < x < 1000$), citită de la tastatură, reprezintă un număr perfect (egal cu suma divizorilor săi, strict mai mici decât el). Vor fi definite și apelate două subprograme:

- funcția D , care determină suma divizorilor unei valori transmise printr-un parametru întreg k , divizori strict mai mici decât k .
- funcția F , care determină numărul de valori perfecte dintr-un interval $[a, b]$ transmis prin intermediul a doi parametri întregi ($1 < a < b$).

Exemplu: Pentru $x=28$ programul va afișa mesajul "Da" ($28=1+2+4+7+14$), iar pentru $x=29$ se va afișa mesajul "Nu".

Soluție:

Funcția D va fi apelată în cadrul subprogramului F , pentru fiecare din valorile întregi din intervalul $[a, b]$, contorizându-se numărul de valori perfecte.

Condiția ca numărul x să fie perfect este echivalentă cu expresia $F(2, x-1) \neq F(2, x)$. În cadrul ambelor funcții se efectuează doar transfer prin valoare.

```
1  var x: word;                       # include<iostream.h>
2                                     int x;
3  function D(k:integer):word;        long int D(int k)
4  var s,i:word;                      {long int s,i;
5  begin                               s=0;
6  s:=0;                               for (i=1;i<=k/2;i++)
7  for i:=1 to k div 2 do               if (k%i==0) s+=i;
8  if k mod i=0 then s:=s+i;           return s;
9  D:=s;                               }
10 end;
11
12 function F(a,b:integer):word;      int F(int a,int b)
13 var s,i:integer;                   {int s,i;
14 begin                               s=0;
15 s:=0;                               for (i=a;i<=b;i++)
16 for i:=a to b do                    if (D(i)==i) s++;
17 if D(i)=i then inc(s);              return s;
18 F:=s;                               }
19 end;
20
21 void main()
22 {
23 readln(x);
24 if F(2, x-1)<>F(2, x) then
25   write('DA')
26 else write('NU');
27 }
```

14. Considerăm un vector a cu n ($n < 100$) elemente numere naturale distincte (< 9999) și o valoare k naturală ($k < n$). Se dorește determinarea celui de-al k -lea element din vectorul ordonat. Subprogramul P determină acest element fără a sorta efectiv elementele vectorului. Acesta are ca parametri pe a , n și k , iar valoarea determinată o returnează prin intermediul unui parametru întreg x .

În cadrul subprogramului se face apel la funcția Nr care determină câte elemente dintr-un vector sunt mai mici decât o valoare dată. Vectorul, numărul de elemente ale acestuia și valoarea respectivă sunt transmise prin parametri.

Scrieți definițiile complete ale subprogramelor P și Nr .

Exemplu: Pentru tabloul ce conține 5 elemente: (220, 45, 5125, 102, 53) și $k=3$, se va determina valoarea 102.

Soluție:

În varianta Pascal, subprogramul P va fi implementat ca procedură, deoarece rezultatul este returnat printr-un parametru. Din același motiv, în C++, funcția P are tipul `void`.

Subprogramul va identifica prin apeluri la funcția Nr , elementul din vector pentru care se regăsesc $k-1$ elemente de valori strict mai mici decât el.

```

1 function Nr(a:sir;
2   n,x:integer):byte;
3 var t,i:integer;
4 begin
5   t:=0;
6   for i:=1 to n do
7     if a[i]<x then inc(t);
8   Nr:=t;
9 end;

10
11 procedure P(a:sir;n,k:integer;
12   var x:integer);
13 var i:integer;
14 begin
15   i:=1;
16   while (Nr(a,n,a[i])<>k-1) and
17     (i<=n) do inc(i);
18   x:=a[i];
19 end;

int Nr(int a[100],int n,int x)
{
  int t,i;
  t:=0;
  for (i=0;i<n;i++)
    if (a[i]<x) t++;
  return t;
}

void P(int a[100],int n,
  int k,int &x)
{
  int i;
  i=0;
  while ((Nr(a,n,a[i])!=k-1)&&
    (i<=n)) i++;
  x=a[i];
}

```

15. Realizați un program care descompune un număr natural n ($3 \leq n \leq 250$), în sumă de minimum doi termeni distincți ai șirului lui Fibonacci ($f_1=1, f_2=1, f_3=2, f_n=f_{n-1}+f_{n-2}$).

În cadrul programului va fi definită o funcție *Fib* care determină cel mai mare termen din șirul Fibonacci, strict mai mic decât o valoare naturală, transmisă prin intermediul unui parametru.

Exemplu: Pentru $n=21$ se va afișa $13+5+2+1$, iar pentru $n=150$ se va afișa $144+5+1$

Soluție:

Funcția *Fib* va avea un singur parametru, transmis prin valoare, și va returna o valoare dintr-un tip întreg. Ea va fi apelată inițial pentru valoarea n , apoi pentru diferența dintre n și valoarea returnată anterior, ș.a.m.d. Toate valorile obținute în urma apelurilor la *Fib*, vor reprezenta termeni din scrierea lui n ca sumă de elemente ale șirului lui Fibonacci.

```

1 var n,x:byte;
2
3 function fib(n:byte):byte;
4 var a,b,c:byte;
5 begin
6   a:=1;b:=1;
7   while a+b<n do begin
8     c:=a+b;
9     a:=b;
10    b:=c;
11  end;
12  fib:=b;
13 end;
14
#include <iostream.h>
int n,x;

int fib(int n)
{int a,b,c;
a:=1; b:=1;
while (a+b<n) {
  c:=a+b;
  a:=b;
  b:=c;
}
return b;
}

```

```

15 begin
16   readln(n);
17   while n>2 do begin
18     x:=fib(n);
19     write(x,' ');
20     n:=n-x;
21   end;
22   writeln(n)
23 end.

void main()
{cin >>n;
while (n>2){
  x=fib(n);
  cout<<x<<' ';
  n=n-x;
}
cout<<n<<' ';
}

```

16. Se consideră un șir de n numere scrise în baza 16. Să se realizeze un program care determină cel mai mare divizor comun al acestor numere, afișându-l în reprezentarea din baza 10. În cadrul programului, vor fi definite și apelate două subprograme:

- funcția *Conv*, care primește un șir de caractere reprezentând un număr exprimat în baza 16 și returnează numărul obținut în urma conversiei acestuia în baza 10.
- funcția *Cmmdc*, care returnează cel mai mare divizor comun a două valori naturale transmise prin intermediul a doi parametri.

Cifrele din baza 16 sunt reprezentate prin cifrele zecimale 0,...,9 și majusculile A,B,C,D,E,F (corespunzătoare resturilor de la 10 la 15).

Exemplu: Pentru $n=3$ și numerele 1A, 27 și 41 afișează valoarea 13.

Soluție:

Ambele funcții vor realiza transfer de date prin intermediul unor parametri valoare. Funcția *Conv* va fi apelată pentru fiecare număr hexazecimal. Valorile returnate de ea vor constitui parametri actuali (efectivi) în cadrul apelurilor la funcția *Cmmdc*.

```

1 var s:string; n,i,a,b:integer;
2
3 function Conv(s:string):integer;
4 var i,nr:integer;
5 begin
6   nr:=0;
7   for i:=1 to length(s) do
8     if s[i] in ['0'..'9'] then
9       nr:=nr*16 + ord(s[i])-48
10    else
11      nr:=nr*16 + ord(s[i])-55;
12   Conv:=nr;
13 end;
14
15 function Cmmdc(a,b:integer):integer;
16 begin
17   while a<>b do
18     if a>b then a:=a-b
19     else b:=b-a;
20   cmmdc:=a;
21 end;
22
#include <iostream.h>
#include <string.h>
char s[100]; int n,i,a,b;

int Conv(char s[100])
{int i,nr:=0;
for (i=0;i<strlen(s);i++)
if (s[i]>='0' && s[i]<='9')
  nr:=nr*16 + s[i]-48;
else
  nr:=nr*16 + s[i]-55;
return nr;
}

int Cmmdc(int a,int b)
{
while (a!=b)
if (a>b) a:=a-b;
else b:=b-a;
return a;
}

```



```

23 begin
24   readln(n); readln(s);
25   a:=Conv(s);
26   for i:=2 to n do begin
27     readln(s); b:=Conv(s);
28     a:=Cmmdc(a,b);
29   end;
30   writeln(a);
31 end.

```

```

void main(){
  cin>>n; cin>>s;
  a=Conv(s);
  for (i=2; i<=n ;i++){
    cin >>s; b=Conv(s);
    a=Cmmdc(a,b);
  }
  cout<<a ;
}

```

17. Considerăm o matrice pătratică a cu n linii și n coloane ($n \leq 100$). Se dorește ordonarea elementelor diagonalei principale prin interschimbări de linii și coloane. Scrieți definițiile următoarelor subprograme:

- subprogramul *interl*, care interschimbă elementele a două linii, ale căror indici sunt transmiși ca parametri.
- subprogramul *interc*, care interschimbă elementele a două coloane, ale căror indici sunt transmiși ca parametri.
- subprogramul *S* care ordonează elementele diagonalei secundare prin apeluri la *interl* și *interc*.

Toate cele trei subprograme vor avea matricea a și numărul n de linii transmise prin parametri.

Exemplu: Pentru $n=3$ și tabloul

3 2 1	1 3 2
2 1 3	1 2 3
3 1 2	2 1 3

se va afișa tabloul

Soluție:

În varianta Pascal, toate subprogramele vor fi implementate ca proceduri având ca parametru referință tabloul bidimensional. Din același motiv, în C++ toate cele trei funcții au tipul *void*, numele tabloului transmis ca parametru fiind el însuși un pointer.

Pentru varianta Pascal considerăm definit tipul de date *mat=array[1..100,1..100] of byte*.

```

1 procedure interl(var a:mat;
2                   n,x,y:byte);
3 var i,t:byte;
4 begin
5   for i:=1 to n do begin
6     t:=a[x,i];
7     a[x,i]:=a[y,i];
8     a[y,i]:=t;
9   end;
10 end;
11
12 procedure interc(var a:mat;
13                   n,x,y:byte);
14 var i,t:byte;

```

```

void interl(int a[100][100],
            int n,int x,int y)
{
  int i,t;
  for (i=0;i<n;i++){
    t=a[x][i];
    a[x][i]=a[y][i];
    a[y][i]=t;
  }
}

void interc(int a[100][100],
            int n,int x,int y)
{
  int i,t;

```

```

15 begin
16   for i:=1 to n do begin
17     t:=a[i,x]; a[i,x]:=a[i,y];
18     a[i,y]:=t;
19   end;
20 end;
21
22 procedure S(var a:mat;n:byte);
23 begin
24   for i:=1 to n-1 do
25     for j:=i+1 to n do
26       if a[i,i]>a[j,j] then begin
27         interl(a,n,i,j);
28         interc(a,n,i,j);
29       end;
30 end;

```

```

for (i=0;i<n;i++){
  t=a[i][x]; a[i][x]=a[i][y];
  a[i][y]=t;
}

void S(int a[100][100],int n)
{
  for (i=0;i<n-1;i++)
    for (j=i+1;j<n;j++)
      if (a[i][i]>a[j][j])
        {
          interl(a,n,i,j);
          interc(a,n,i,j);
        }
}

```

18. Considerăm o funcție *S* care returnează suma tuturor cifrelor a două valori naturale transmise prin intermediul a doi parametri.

Se dorește determinarea unei perechi de elemente distincte, ale unui tablou unidimensional care au proprietatea că suma tuturor cifrelor lor este maximă.

Vectorul se consideră a avea maximum 100 de elemente pozitive cu valori mai mici decât 2.000.000.000.

Subprogramul *P*, primește prin doi parametri, vectorul și numărul lui de elemente și returnează, prin intermediul altor doi parametri, elementele tabloului ce au proprietatea cerută. Acesta va face apel la funcția *S*.

Scrieți definițiile celor două subprograme.

Exemplu: Pentru $n=5$ și tabloul (86, 15, 13, 86, 112), subprogramul *P* va returna valorile 86 și 15 ($8+6+1+5=20$ =suma maximă).

Soluție:

În varianta Pascal, subprogramul *P* va fi implementat ca procedură, având doi parametri referință întregi. Aceștia vor returna elementele tabloului, pentru care suma tuturor cifrelor este maximă. Din același motiv, în C++ funcția *P* are tipul *void*.

Tabloul și numărul de elemente ale acestuia vor reprezenta, pentru subprogramul *P*, parametri valoare. Pentru varianta Pascal considerăm definit tipul de date *sir=array[1..100] of longint*.

```

1 function S(x,y:longint):byte;
2 var su:byte;
3 begin
4   su:=0;
5   while (x>0) or (y>0) do begin
6     su:=su+x mod 10+y mod 10;
7     x:=x div 10; y:=y div 10;
8   end;
9   S:=su;
10 end;

```

```

int S(long x,long y)
{
  int su;
  su:=0;
  while (x>0 || y>0){
    su += x % 10 + y % 10;
    x /= 10; y /= 10;
  }
  return su;
}

```

```

11 procedure P(a:sir;n:byte;
12             var x,y:longint);
13 var max,i,j:byte;
14 begin
15   max:=0;
16   for i:=1 to n-1 do
17     for j:=i+1 to n do
18       if (max<S(a[i],a[j]))and
19         (a[i]<>a[j]) then begin
20         max:=S(a[i],a[j]);
21         x:=a[i]; y:=a[j];
22       end;
23   end;

```

```

void P(long a[100],int n,
      long & x,long & y)
{int max,i,j;
  max:=0;
  for (i=0;i<n-1;i++)
    for (j=i+1;j<n;j++)
      if (max<S(a[i],a[j])
        && a[i]!=a[j])
        {
          max=S(a[i],a[j]);
          x=a[i]; y=a[j];
        }
}

```

19. Fie un tablou unidimensional cu n elemente întregi. Se cere să se realizeze un subprogram S care ordonează descrescător elementele vectorului, după numărul de cifre distincte pe care le conțin. În cazul elementelor cu același număr de cifre distincte, ordonarea se va face descrescător după valorile lor. Subprogramul va apela funcția Nr care determină numărul de cifre distincte ale unui număr întreg primit prin intermediul unui parametru.

Exemplu: Primind $n=7$ și tabloul (334, 124, 21, 34, 222, 1, 9) subprogramul S va transmite, în urma execuției, tabloul (124, 334, 34, 21, 222, 9, 1).

Soluție:

În varianta Pascal, subprogramul S va fi implementat ca procedură, având ca parametru referință tabloul unidimensional și ca parametru valoare numărul de elemente ale acestuia. Din același motiv, în C++ funcția S are tipul *void*, numele tabloului transmis ca parametru fiind el însuși un pointer.

Pentru varianta Pascal considerăm definit tipul de date $sir=array[1..100]$ of integer.

```

1 function Nr(x:integer):byte;
2 var m:byte; c:set of byte;
3 begin
4   m:=0;
5   c:={};
6   while (x>0) do begin
7     if not (x mod 10 in c) then
8       begin
9         inc(m);
10        c:=c+[x mod 10];
11      end;
12     x:=x div 10;
13   end;
14   Nr:=m;
15 end;

```

```

1 int Nr(int y)
2 {
3   int x,i,m,c;
4   m:=0;
5   for (i=0;i<10;i++){
6     c:=0;
7     x=y;
8     while (x>0) {
9       if (x % 10 == i) c=1;
10      x /= 10;
11    }
12    if (c) m++;
13  }
14  return m;
15 }

```

```

17 procedure S(var a:sir;n:byte);
18 var i,j:byte;x:integer;
19 begin

```

```

void S (int a[100], int n)
{
  int i,j,x;

```

```

20 for i:=1 to n-1 do
21   for j:=i+1 to n do
22     if (Nr(a[i])<Nr(a[j]))or
23       ((Nr(a[i])=Nr(a[j])) and
24         (a[i]<a[j]))
25     then begin
26       x:=a[i]; a[i]:=a[j];
27       a[j]:=x;
28     end;
29 end;

```

```

for (i=0;i<n-1;i++)
  for (j=i+1;j<n;j++)
    if (Nr(a[i])<Nr(a[j]) ||
      (Nr(a[i])=Nr(a[j]) &&
        a[i]<a[j]))
    {
      x=a[i]; a[i]=a[j]; a[j]=x;
    }
}

```

20. Se consideră un vector ce conține n elemente întregi. În fața oricărui element precedat de un element de semn contrar se inserează un element pozitiv, a cărui valoare este obținută prin alipirea cifrelor celor două numere de semne contrare, în ordine. Să se scrie definițiile următoarelor trei subprograme:

- funcția L , care returnează numărul natural obținut prin alipirea, în ordine, a cifrelor a două valori primite prin intermediul unor parametri întregi.
- subprogramul Ins , care permite inserarea într-un vector, pe o poziție dată, a unei valori date. Vectorul, indicele pe care urmează să se efectueze inserarea și valoarea care se va insera sunt transmise subprogramului prin intermediul parametrilor.
- subprogramul Rez , care în urma apelurilor la subprogramele Ins și A , efectuează prelucrarea cerută prin enunț, asupra unui vector pe care îl primește pîntr-un parametru.

Exemplu: După execuția subprogramului Rez , tabloul (3,-1, 73, 5, -9, 2) va conține elementele (3, 31,-1, 173, 73, 5, 59, -9, 92, 2).

Soluție:

În varianta Pascal, subprogramele Ins și Rez vor fi implementate ca proceduri, ambele având transmiși, prin parametri referință, tabloul unidimensional și lungimea acestuia. În C++, funcțiile Ins și Rez au tipul *void*, transferul tabloului și a lungimii acestuia realizându-se tot prin parametri referință.

Funcția L va fi apelată pentru oricare pereche de elemente consecutive de semne contrare. Valorile returnate de ea, vor constitui parametri actuali la apelul subprogramului Ins .

Pentru varianta Pascal, considerăm definit tipul de date $sir=array[1..100]$ of integer.

```

1 function L(x,y:integer):integer;
2 var z:integer;
3 begin
4   z:=y;
5   while (z>0) do begin
6     x:=x*10;
7     z:=z div 10;
8   end;
9   L:=x+y;
10 end;

```

```

1 int L(int x,int y)
2 {
3   int z;
4   z=y;
5   while (z>0) {
6     x *=10;
7     z/=10;
8   }
9   return x+y;
10 }

```

```

11 procedure Ins(var a:sir;
12     var n:byte; x,p:integer);
13 var i,j:byte;
14 begin
15     for i:=n downto p do
16         a[i+1]:=a[i];
17     a[p]:=x; inc(n)
18 end;
19
20 procedure Rez(var a:sir;
21     var n:byte);
22 var x,y:integer;
23 begin
24     i:=1;
25     while i<n do
26         if a[i]*a[i+1]<0 then begin
27             x:=abs(a[i]); y:=abs(a[i+1]);
28             Ins(a,n,L(x,y),i+1);
29             inc(i,2);
30         end
31         else inc(i)
32     end;

```

```

void Ins(int a[100],
    int &n,int x,int p)
{
    int i,j;
    for (i=n-1;i>=p;i--)
        a[i+1]=a[i];
    a[p]=x; n++;
}

void Rez(int a[100], int &n)
{
    int x,y;
    i=0;
    while (i<n-1)
        if (a[i]*a[i+1]<0){
            x=abs(a[i]); y=abs(a[i+1]);
            Ins(a,n,L(x,y),i+1);
            i += 2;
        }
        else i++;
}

```

2.1.3 Probleme propuse

1. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere naturale. Se cere realizarea unui program care înlocuiește fiecare element din vector cu cel mai apropiat număr prim față de acesta. Programul va conține următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Prim*, care verifică dacă valoare primită printr-un parametru întreg este număr prim. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- Funcția *Det*, care va determina cel mai apropiat număr prim față de o valoare întreagă primită printr-un parametru, folosindu-se de apeluri la funcția *Prim*. Rezultatul funcției *Det* va fi de tip întreg.
- subprogramul *Scrie*, care permite afișarea elementelor unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=4$ și tabloul $a=(5, 16, 33, 24)$ se va afișa: 5 17 31 23

2. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere întregi. Să se realizeze un program care ordonează crescător elementele situate în prima jumătate a vectorului și descrescător pe cele situate în a doua jumătate. Programul va conține următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- subprogramul *Sortare*, care ordonează la alegere elementele unui vector, între doi indici transmiși prin intermediul a doi parametri întregi. (vezi problema 5, secțiunea 2.1.2). Vectorul va fi transmis subprogramului printr-un parametru.
- subprogramul *Scrie*, care permite afișarea elementelor unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=5$ și tabloul $a=(153, 16, 8, 33, 124)$ se va afișa:

16 153 124 33 8

3. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere întregi. Să se realizeze un program care identifică un subtablou a lui a , format din k elemente, subtablou care conține cele mai multe valori cuburi perfecte.

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor variabilelor n , k și a elementelor vectorului a .
- funcția *Cub*, care verifică dacă valoare primită printr-un parametru întreg este cub perfect. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- funcția *Poz*, care determină poziția (indicele) de început a subtabloului de k elemente care conține cele mai multe cuburi perfecte. Funcția va face apel la subprogramul *Cub*. Vectorul, numărul de elemente ale acestuia și valoarea lui k vor fi primite de subprogram prin intermediul parametrilor.
- subprogramul *Scrie*, care permite afișarea unei secvențe de k elemente, începând cu o poziție p , dintr-un vector transmis prin parametru. Valorile lui k și p vor fi primite de subprogram prin intermediul a doi parametri întregi.

Exemplu: Pentru $n=7$, $k=4$ și tabloul $a=(1, 9, 4, 27, 8, 1, 12)$ se va afișa: 4, 27, 8, 1

4. Se consideră două numere naturale, n și b ($n \leq 100$, $b \leq 10$). Să se construiască un tablou unidimensional ale cărui elemente vor fi primele n numere naturale, care au proprietatea că în baza b se reprezintă numai cu cifre de 0 și 1.

În cadrul programului, vor fi definite următoarele subprograme:

- funcția *Baza*, care verifică dacă valoare primită printr-un parametru întreg se reprezintă într-o bază b doar prin cifre de 0 și 1. Baza b va fi primită tot prin intermediul unui parametru întreg. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Det*, care construiește un vector cu n elemente. Acestea reprezintă primele n valori naturale ce au proprietatea că în baza b sunt scrise numai cu cifre de 0 și 1. Tabloul, valoarea lui n și a lui b sunt transmise subprogramului prin intermediul parametrilor.

- subprogramul *Scrie*, care permite afișarea elementelor unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=7$ și $b=3$ tabloul va conține valorile 0, 1, 3, 4, 9, 10, 12.

5. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere naturale, mai mici ca 2.000.000.000. Să se realizeze un program care afișează pe linii, în fișierului *Nr.txt*, elementele din vectorul a , grupate după cifra-dominantă (prima în scrierea zecimală). O linie va conține doar elemente cu aceeași cifră dominantă. Scrierea în fișier se va face în ordinea crescătoare a primei cifre.

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Cif*, care determină cifra dominantă a unei valori primite printr-un parametru întreg.
- subprogramul *Scrie*, care afișează pe linii, în fișierul *Nr.txt*, elementele din a , după regula prezentată în enunț. Se vor folosi apeluri la funcția *Cif*. Tabloul și numărul de elemente ale acestuia sunt transmise subprogramului prin intermediul parametrilor.

Exemplu: Pentru $n=7$ și tabloul $a=(334, 124, 71, 34, 122, 1, 39)$ fișierul *Nr.txt* va conține:

```
124 122 1
334 34 39
71
```

6. Se consideră două tablouri unidimensionale a și b ce conțin n , respectiv m elemente numere întregi ($m \leq n \leq 50$). Să se realizeze un program care determină de câte ori toate elementele lui b apar pe poziții consecutive în tabloul a .

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a numărului de elemente ale unui vector și a elementelor acestuia. Transferul datelor citite va fi realizat prin parametri.
- funcția *Ok*, care verifică dacă elementele unui vector b se regăsesc în vectorul a , pe poziții consecutive, începând de la un indice dat. Funcția va avea 5 parametri: cele două tablouri, numărul de elemente ale fiecăruia și indicele începând cu care se face verificarea. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- funcția *Nr*, care determină numărul de apariții cerut în enunț. Cei doi vectori și numărul de elemente ale fiecăruia sunt transmise subprogramului prin intermediul parametrilor. Se vor folosi apeluri la funcția *Ok*.

Exemplu: Pentru $n=7$, $m=3$ și tablourile $a=(33, 3, 71, 3, 71, 3, 39)$, $b=(3, 71, 3)$, funcția *Nr* va returna valoarea 2.

7. Se consideră un tablou unidimensional a cu n ($n < 100$) elemente întregi. Să se determine toate secvențele de elemente situate pe poziții consecutive care au suma egală cu S . Fiecare secvență de elemente va fi afișată pe câte o linie pe ieșirea standard.

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor variabilelor n , S și ale elementelor vectorului a .
- funcția *Sum*, care determină suma elementelor unui vector, situate între două poziții date. Tabloul și cei doi indici vor fi transmiși subprogramului prin intermediul parametrilor.
- subprogramul *Scrie*, care permite afișarea tuturor secvențelor de elemente dintr-un tablou care respectă cerința din enunț. Vectorul și numărul de elemente ale acestuia vor fi primiți de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=7$, $S=9$ și vectorul $(3, 2, 3, 4, 5, 11, -7)$ se va afișa:

```
2 3 4
4 5
5 11 -7
```

8. Se consideră un tablou unidimensional a ce conține n ($n \leq 100$) elemente numere naturale, mai mici ca 20.000. Să se realizeze un program care determină un subtablou al său, cu proprietatea că suma elementelor este divizibilă cu n .

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Rest*, care determină restul la împărțirea cu n a sumei primelor k elemente ale unui vector. Tabloul, n și k vor fi trimise subprogramului prin intermediul parametrilor.
- subprogramul *Det*, determină doi indici din vectorul a , care delimitează subtabloul cu proprietatea prezentată în enunț. Se vor folosi apeluri la funcția *Rest*. Subprogramul va primi, prin intermediul a doi parametri, vectorul și numărul de elemente ale acestuia și va returna, prin intermediul altor doi parametri întregi, cei doi indici determinați.
- subprogramul *Scrie*, care permite afișarea unei secvențe de elemente dintr-un tablou, situate între două poziții date. Vectorul și cei doi indici vor fi primiți de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=7$ și tabloul $a=(4, 5, 1, 3, 2, 0, 9)$, se va afișa 3 2 0 9.

9. Se consideră două mulțimi reținute în doi vectori. Să se realizeze un program care determină reuniunea, intersecția și diferența lor. Pentru fiecare dintre operații se cere definirea unui subprogram. Acestea vor apela funcția *Ap* care verifică dacă o valoare dată aparține unei mulțimi. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++. Transferul datelor între subprograme va fi realizat prin intermediul parametrilor.

Exemplu : $A=(2, 4, 1, 6, 7)$, $B=(3, 4, 8, 9)$ se va afișa:

Reuniunea= $(2, 4, 1, 6, 7, 3, 8, 9)$; Intersectia= (4) ; Diferenta= $(2, 1, 6, 7)$

10. Considerăm un vector a ce reprezintă o permutare a mulțimii numerelor de la 1 la n ($n \leq 50$). Se cere determine toate permutările circulare ale lui a care nu conțin puncte fixe.

Spunem că o permutare $(a_1, a_2 \dots a_n)$ are puncte fixe dacă există cel puțin un element $a_i = i$ ($1 \leq i \leq n$). De exemplu, pentru $n=4$, permutarea $(2,4,3,1)$ are puncte fixe, deoarece $a_3=3$.

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor vectorului a .
- funcția *Pfix*, care verifică dacă un vector, ce reprezintă o permutare, are puncte fixe. Vectorul și lungimea acestuia vor fi primite prin intermediul parametrilor. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Perm*, care va genera o permutare circulară cu o poziție spre stânga a unui vector primit printr-un parametru.
- subprogramul *Scrie*, care permite afișarea permutărilor circulare ale unui vector, permutări ce nu conțin puncte fixe. Acest subprogram va face apel la *Pfix* și *Perm*. Tabloul și numărul de elemente ale acestuia sunt primite de subprogramul *Scrie* prin intermediul a doi parametri.

Exemplu: Pentru $n=4$ și tabloul $a=(5, 1, 3, 2, 4)$ se va afișa:

2 4 5 1 3

4 5 1 3 2

11. Se consideră un tablou bidimensional pătratic cu n linii ($n \leq 10$). Să se determine elementele care sunt situate pe linii și coloane de sumă egală. Un element $a[i,j]$ va fi afișat dacă suma pe linia i este egală cu suma pe coloana j .

În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor tabloului a .
- funcția *SumL*, care calculează suma elementelor dintr-o matrice, situate pe o linie, al cărui număr de ordine este transmis printr-un parametru întreg.
- funcția *SumC*, care calculează suma elementelor dintr-o matrice, situate pe o coloană, al cărui număr de ordine este transmis printr-un parametru întreg.
- subprogramul *Scrie*, care permite afișarea tuturor elementelor dintr-o matrice ce respectă cerința din enunț. Tabloul și numărul de linii/coloane, sunt primite de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=4$ și tabloul

0 5 0 0

3 7 2 4

-20 6 20 10

0 -2 0 0

se vor afișa elementele 7 și 6.

12. Se consideră un tablou bidimensional pătratic cu n linii ($n \leq 20$). Să se determine toate elementele ce reprezintă puncte 'șă' (element minim pe linie și maxim pe coloana pe care este situat).

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor tabloului a .
- funcția *MinL*, care determină cea mai mică valoare situată în matricea a pe o linie, al cărui număr de ordine este transmis printr-un parametru întreg.
- funcția *MaxC*, care primește două valori întregi, prin intermediul a doi parametri x și c . Ea verifică dacă x reprezintă maximum pe coloana c a matricii a . Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Scrie*, care permite afișarea tuturor punctelor „șă” dintr-o matrice. Tabloul și numărul de linii/coloane, sunt primite de subprogram prin intermediul parametrilor.

Exemplu: Pentru $n=4$ și tabloul

9 8 6 7

8 3 5 4

9 9 6 7

5 1 4 3

se vor afișa elementele 6 și 6.

13. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente întregi. Să se identifice numerele de ordine ale coloanelor care conțin cele mai multe elemente în afara intervalului $[x,y]$.

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n și m și ale elementelor tabloului a .
- funcția *Nr*, care primește trei valori întregi prin intermediul parametrilor x , y și c . Ea determină câte elemente situate pe coloana c , în matricea a , se află în afara intervalului $[x,y]$.
- subprogramul *Det*, care afișează indicii coloanelor cu proprietatea cerută în enunț. În cadrul acestuia se va face apel la funcția *Nr*.

Exemplu: Pentru $n=4$, $m=5$, $x=2$, $y=7$ și tabloul

2 8 5 8 3

4 3 6 7 5

8 8 8 8 8

9 9 9 9 9

se va afișa 2 4 (coloana a doua și a patra).

14. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$) cu componente întregi. Folosind interschimbări de coloane, să se ordoneze crescător elementele pare situate pe ultima linie din tabloul a .

În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n și m și ale elementelor tabloului a .
- subprogramul *InterC*, care primește două valori întregi prin intermediul parametrilor x, y . Acesta interchimbă elementele coloanei x cu cele situate pe coloana y în matricea a .
- subprogramul *Sortare*, care efectuează ordonarea crescătoare a elementelor pare situate pe ultima linie în matricea a folosind apeluri la *InterC*.
- subprogramul *Scrie*, care efectuează afișarea pe ecran a elementelor tabloului bidimensional a .

Exemplu: Pentru $n=3, m=4$ și tabloul a :

1 2 3 4	se va afișa:
1 2 3 4	1 4 2 3
1 4 6 2	1 4 2 3
	1 2 4 6

15. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente întregi. Să se identifice numerele de ordine ale liniilor care conțin elemente ordonate crescător și, în plus, diferența dintre oricare două elemente alăturate, este constantă în cadrul liniei. În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Ok*, care primește o valoare naturală prin intermediul parametrului i și verifică dacă elementele liniei de indice i , din matricea a , respectă regula din enunț. Rezultatul funcției va fi de tip logic în varianta Pascal și de tip întreg (0/1) pentru C++.
- subprogramul *Scrie*, care efectuează afișarea pe ecran a numerelor de ordine determinate.

Exemplu: Pentru $n=4$ și tabloul a :

1 3 5 7
4 3 2 1
2 4 6 8
9 9 9 8

se va afișa 1 3 (prima linie și a treia).

16. Se consideră un tablou bidimensional pătratic a cu n linii și n coloane ($n \leq 50$) cu componente întregi. Să se permută circular cu k poziții, spre dreapta, toate liniile tabloului. În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui k și ale elementelor tabloului a .
- subprogramul *Perm*, care primește o valoare naturală prin intermediul parametrului i și permută toate elementele liniei de indice i , cu o poziție spre dreapta.

- subprogramul *Solve*, care permută cu k poziții spre dreapta, toate liniile unei matrici pătratică, prin apeluri la *Perm*.
- subprogramul *Scrie*, care efectuează afișarea pe ecran a numerelor de ordine ale liniilor lui a care au proprietatea impusă prin enunț.

Tabloul bidimensional pătratic și numărul de linii ale acestuia sunt transmise subprogramelor prin intermediul a doi parametri.

Exemplu: Pentru $n=4, k=2$ și tabloul a :

1 2 3 4	se va afișa:
1 2 3 4	3 4 1 2
1 4 6 2	3 4 1 2
2 4 5 3	6 2 1 4
	5 3 2 4

17. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente numere naturale ≤ 60000 . Să se afișeze, pentru fiecare linie, numărul de zerouri în care se termină produsul elementelor lor. În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Exp*, care returnează exponentul la care apare un număr prim în descompunerea unui număr natural. Ambele valori sunt primite de subprogram prin intermediul a doi parametri întregi.
- funcția *Nrz*, care primește o valoare naturală prin intermediul parametrului i și determină numărul de zerouri în care se termină produsul elementelor situate pe linia i , în matricea a . Se vor folosi apeluri la funcția *Exp*.
- subprogramul *Scrie*, care efectuează afișarea pe ecran, pentru fiecare linie, a numărului de zerouri în care se termină produsul elementelor lor. Se vor folosi apeluri la funcția *Nrz*.

Exemplu: Pentru $n=4, m=3$ și tabloul a

2 8 5
4 3 6
10 10 10
25 25 4

se va afișa 1, 0, 3, 2.

18. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente cifre zecimale (0..9). Considerăm că fiecare linie reprezintă cifrele unui număr. Se cere identificarea celei mai mici baze comune tuturor numerelor reprezentate în matrice și afișarea acestora în urma conversiei din baza minimă determinată, în baza 10.

În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Baza*, care determină cea mai mică bază comună tuturor numerelor reprezentate în matricea a .

- funcția *Conv*, care primește două valori naturale prin intermediul parametrilor b și l . Ea determină valoarea din baza 10 obținută în urma conversiei numărului reprezentat în baza b pe linia l .
- subprogramul *Scrie*, care efectuează afișarea pe ecran, pentru fiecare linie, a numărului returnat în urma apelului la funcția *Conv*.

Exemplu: Pentru $n=3$, $m=3$ și tabloul a

```
2 1 5
4 3 6
1 0 1
```

se va afișa 110, 223, 50 (baza minimă 7).

19. Se consideră un tablou bidimensional a cu n linii și m coloane ($1 \leq n, m \leq 50$), cu componente cifre zecimale (0..9). De la tastatură este introdus un șir de $m-1$ caractere $+$ și $-$, având semnificația operatorilor aritmetici cunoscuți. Ele vor fi "plasate" în ordine între valorile de pe fiecare linie. Se cere să se determine care este valoarea maximă care se obține în urma evaluării expresiilor obținute în cadrul fiecărei linii.

În cadrul programului, se vor defini subprogramele următoare:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale elementelor tabloului a .
- funcția *Eval*, care primește o valoare naturală printr-un parametru l și un șir de caractere printr-un parametru s . Aceasta evaluează expresia obținută prin plasarea în ordine a operatorilor $+$ și $-$ din care este format s , între elementele situate pe linia l a matricei a .
- funcția *Max*, care determină valoarea maximă care se obține în urma evaluării expresiilor obținute în cadrul fiecărei linii, prin apeluri la funcția *Eval*.

Exemplu: Pentru $n=3$, $m=4$, $s=,+-$ și tabloul a

```
2 1 5 2
4 3 6 3
```

1 0 1 2 se va afișa 10 (4+3+6-3)

20. Se consideră un număr natural n ($n \leq 100$). Să se creeze o matrice ale cărei elemente sunt numerele de la 1 la n^2 . Valorile sunt plasate în matrice, consecutiv în cadrul unei linii, dar alternând monotonia, după cum reiese și din tabloul următor:

```
1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13
```

În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *linieC*, care primește două valori naturale prin intermediul parametrilor x și l . Acesta plasează, crescător, valori consecutive pe linia l începând cu valoarea x .

- subprogramul *linieD*, care primește două valori naturale prin intermediul parametrilor x și l . Acesta plasează, descrescător, valori consecutive pe linia l începând cu valoarea x .
- subprogramul *Solve*, care construiește o matrice pătratică de ordin n , ale cărei elemente respectă regula din enunț. Acesta va face apel la subprogramele *linieC* și *linieD*. Matricea și numărul n de linii este transmis subprogramului *Solve* prin intermediul a doi parametri.
- subprogramul *Scrie*, care efectuează afișarea pe ecran elementelor unui tablou bidimensional pătratic transmis printr-un parametru. Numărul de linii al său este transmis prin al doilea parametru.

21. Se consideră un șir de cel mult 100 de caractere, format din litere mici ale alfabetului englez. Se numește bălbă o secvență de caractere care apare în șir de cel puțin două ori una după alta. De exemplu șirul "abddcabddabcabc" conține bălbele "d" și "abc".

- Să se realizeze funcția *MaxB*, care determină cea mai lungă "bălbă" dintr-un șir de caractere primit printr-un parametru.
- Să se realizeze funcția *Cod*, care codifică un șir de caractere primit ca parametru. Regula de codificare este următoarea: fiecare caracter este înlocuit cu ultima cifră a numărului 2^x , unde x este codul ASCII asociat literei respective. Rezultatul returnat de funcție este un șir de caractere.

Exemplu: Pentru șirul "abbcabab", funcția *MaxB* va returna șirul "ab", iar funcția *Cod* va returna șirul de caractere "24482424"

22. Considerăm un vector a , care conține n elemente numere reale ($n \leq 100$). Se dorește construirea unui vector b cu n elemente numere întregi, creat pe baza elementelor lui a după următoarea regulă: dacă elementul $a[i]$ are partea întreagă un număr par, atunci $b[i]$ va reprezenta cel mai apropiat întreg de $a[i]$ divizibil cu 10; în caz contrar, $b[i]$ va fi egal cu cel mai apropiat întreg de $a[i]$ divizibil cu 100. În cadrul programului, vor fi definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorii lui n și a elementelor reale ale vectorului a .
- funcția *Div10*, care rotunjește o valoare reală, primită printr-un parametru, la cel mai apropiat întreg divizibil cu 10.
- funcția *Div100*, care rotunjește o valoare reală, primită printr-un parametru, la cel mai apropiat întreg divizibil cu 100.
- subprogramul *Solve*, care primind prin parametrul a un vector cu elemente reale, returnează printr-un alt parametru b , un vector ale cărui elemente întregi respectă regula din enunț. Se vor folosi apeluri la funcțiile *Div10* și *Div100*.
- subprogramul *Scrie* care permite afișarea elementelor întregi ale unui vector. Tabloul și numărul de elemente ale acestuia sunt primite de subprogram prin intermediul a doi parametri.

Exemplu: Pentru $n=3$ și $a=(1327.3, 234.67, 487.34)$, se va afișa 1300, 230, 500.

23. Se consideră un tablou bidimensional pătratic a cu n linii ($n \leq 50$), cu componente numere întregi. Să se identifice o zonă pătratică în matrice, cu latura strict mai mică decât n , care conține cele mai multe valori egale cu numărul întreg k . Dacă există mai multe astfel de zone se va afișa cea de arie minimă. Programul va conține definite următoarele subprograme:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui k și ale elementelor tabloului a .
- funcția *SubT*, care primește trei valori întregi prin parametrii x, y, l . Aceasta determină numărul de valori egale cu k din zona care are colțul stânga-sus în coordonatele x, y și este formată din l linii și l coloane.
- subprogramul *Solve*, care determină zona pătratică din matricea a care conține cele mai multe valori egale cu k . Rezultatul este returnat prin trei parametri $x1, y1, lm$ reprezentând coordonatele colțului stânga-sus ($x1, y1$) și numărul lm de linii și coloane ale zonei determinate.

Exemplu: Pentru $n=5, k=3$ și tabloul a

```
2 1 5 6 3
4 3 3 3 3
1 0 1 2 3
2 4 5 6 7
2 3 3 3 6
```

se va afișa 2 2 4 (colțul stânga sus de coordonate 2, 2 având 4 linii și 4 coloane).

24. Numim matrice rară, o matrice ale cărei elemente sunt în majoritate egale cu 0. O astfel de matrice poate fi memorată într-un vector de înregistrări în care fiecare element nenul este memorat împreună cu numărul de ordine al său, obținut în urma liniarizării matricei. Pornind de la un astfel de vector, și cunoscând numărul de linii și de coloane ($n, m \leq 50$) ale matricei rare pe care o codifică, să se genereze în memorie tabloul bidimensional corespunzător acesteia.

Programul va conține definite subprogramele:

- subprogramul *Citeste*, care efectuează citirea de la tastatură a valorilor lui n , lui m și ale tuturor celor nr înregistrări ale vectorului a .
- funcția *L*, care primind trei valori întregi prin parametrii x, y și t , determină linia pe care este situat elementul cu numărul de ordine t , obținut la liniarizarea unei matrice cu x linii și y coloane.
- funcția *C*, care primind trei valori întregi prin parametrii x, y și t , determină coloana pe care este situat elementul cu numărul de ordine t , obținut la liniarizarea unei matrice cu x linii și y coloane.
- subprogramul *Solve*, care construiește în memorie tabloul bidimensional corespunzător matricei rare codificate într-un vector de înregistrări. Șirul este primit ca parametru, împreună cu numărul de elemente al său. Matricea construită este transmisă printr-un alt parametru.

Exemplu: Pentru $n=3, m=4, nr=4$ și vectorul de înregistrări ((2,3) (7,5) (5,6) (4,9) (8,11)) se va construi matricea:

```
0 0 2 0
7 5 0 0
4 0 8 0
```

25. Toate cele n ($n \leq 50$) camere ale unui hotel au formă dreptunghiulară. Pentru mochetarea acestora, patronul apelează la o firmă care îi trimite o ofertă de prețuri și îl asigură de o reducere de $x\%$ din valoarea totală a produselor cumpărate. Patronul dispune de s de lei și se hotărăște să achiziționeze un sortiment de mochetă care costă y lei pe m^2 .

Realizați un program care determină numărul maxim de camere care pot fi mochetae complet, cu suma de bani de care dispune patronul și prețul efectiv pe care acesta îl va plăti. În cadrul programului, vor fi definite subprogramele următoare:

- subprogramul *Citeste*, care preia, de la tastatură, valorile n, x, y, s și n perechi de numere naturale, reprezentând lungimile laturilor fiecărei camere. Subprogramul va returna, prin parametrul a , un vector ale cărui elemente reprezintă șirul ariilor camerelor.
- subprogramul *Ordon*, care efectuează ordonarea crescătoare a elementelor unui vector primit printr-un parametru.
- subprogramul *Calcul*, care primește șirul ariilor prin parametrul a și trei valori întregi prin parametrii s, x și y . Subprogramul returnează, prin parametrul nr , numărul maxim de camere care pot fi mochetae complet cu suma de bani s . Valoarea finală de plată având o reducere de $x\%$ va fi returnată prin parametrul întreg p .

Exemplu:

Pentru $n=4, x=3, y=15, s=1000$ și dimensiunile camerelor (5, 8), (3, 6), (3, 7), (4, 5) se va afișa: 3 camere; Preț final=858.

2.2. Subprograme implementate în manieră recursivă

2.2.1 Teste cu alegere multiplă și duală

1. Considerăm următoarea funcție recursivă:

function F(x:real;n:byte):real;	float F(float x, int n)
begin	{
if n<=2 then F:=2.0	if (n<=2) return 2.0;
else F:=2*x + F(x-1,n-1);	else return 2*x + F(x-1,n-1);
end;	}

Ce valori vor fi afișate după executarea secvenței de instrucțiuni:

```
write(F(3.0,2):0:0,' ');      cout<<F(3.0,2)<<" ";
write(F(2.0,4):0:0,' ');      cout<<F(2.0,4)<<" ";
write(F(2.0,5):0:0,' ');      cout<<F(2.0,5)<<" ";
```

- a) 6 2 2
- b) 2 8 8
- c) 2 5 5
- d) 1 7 7

2. Se consideră următorul program iterativ:

```
procedure invers;
var i,j,t:integer;
    a:array[1..100]of byte;
begin
j:=0;
read(i);
while (trunc(i)<=1) do begin
inc(j); a[j]:=i; read(i);
end;
for t:=j downto 1 do write(a[t])
end;
```

```
void invers()
{int i,j,t,a[100];
j=0;
cin>>i;
while (floor(i)<=1) {
j++;
a[j]=i;
cin>>i;
}
for(t=j;t>=1;t--)cout<<a[t];
}
```

Care dintre subprogramele recursive următoare constituie o variantă echivalentă?

a)

```
procedure invers;
var j:byte;
begin
read(j);
if (j in [0..1]) then begin
invers; write(j);
end;
end;
```

b)

```
procedure invers(j:integer);
begin
read(j);
if (j in [0..1]) then begin
write(j); invers(j);
end;
end;
```

c)

```
procedure invers(i:integer);
var a:array[1..100]of byte;
begin
read(a[i]);
if (a[i] in [0..1]) then
invers(i+1);
write(a[i]);
end;
```

a)

```
void invers()
{int j;
cin>>j;
if (floor(j)<=1){
invers(); cout<<j;
}
}
```

b)

```
void invers(int j)
{
cin>>j;
if (floor(j)<=1){
cout<<j; invers(j);
}
}
```

c)

```
void invers(int i)
{
int a[100];
cin>>a[i];
if (floor(a[i])<=1)
invers(i+1);
cout<<a[i];
}
```

3. La fiecare apel recursiv al unui subprogram, în memoria **STACK** sunt salvate:

- a) adresa de revenire, valorile variabilelor locale și a parametrilor transmiși prin referință;
- b) adresa de revenire și valorile variabilelor globale;
- c) adresa de revenire, valorile variabilelor locale și a parametrilor transmiși prin valoare și adresele parametrilor transmiși prin referință;
- d) adresa de revenire și valorile variabilelor locale și globale.

4. Se consideră programul următor:

```
var x,y:integer;
function f(a,b:word):word;
begin
if a<>b then
if a>b then f:=f(a-b,b)
else f:=f(a,b-a)
else f:= x*y div a;
end;
begin
readln(x,y);
write(f(x,y));
end.
```

```
#include <iostream.h>
int x, y;
long F(int a, int b)
{
if (a!=b)
if (a>b) return F(a-b,b);
else return F(a,b-a);
else return x*y/a;
}
void main()
{
cin>>x>>y ;
cout<<F(x,y);
}
```

Ce valoare va fi afișată pentru setul de date de intrare $x=32, y=14$?

- a) 2
- b) 256
- c) 200
- d) 224
- e) 14

5. Pentru subprogramul recursiv următor, specificați variabilele ale căror valori vor fi reținute în stivă:

```
procedure E(a:byte;var n:byte;
var s:char);
var i,j:integer;
begin
...
if n<100 then E(a,n,s);
...
end;
```

```
void E(int a,int &n, char &s)
{
int i,j;
...
if (n<100) E(a,n,s);
...
}
```

- a) a, n, i, j
- b) n, s, i, j
- c) a, i, j
- d) a, n, s, i, j

6. Care dintre următoarele subprograme permit, ca la executarea lor, să se afișeze cinci valori, dacă parametrul efectiv (la apel) are valoarea 1?

a)
procedure A(i:integer);
begin
 write(i, ' '); A(5*i)
end;

b)
function B(i:integer):integer;
begin
 if i<=10 **then begin**
 write(i, ' '); B:=B(2*i+1)
end else B:=0
end;

c)
function C(i:integer):integer;
begin
 if i<=5 **then begin**
 write(i, ' '); C:=C(i+1); **end**
end;

d)
procedure D(i:integer);
begin
 if i>=-4 **then** D(i-1);
 write(i, ' ');
end;

a)
void A(int i) {
 cout<<i<<" ";
 A(5*i);
}

b)
int B(int i) {
 if (i<=10) {
 cout<<i<<" "; **return** B(2*i+1);
 } **else return** 0;
}

c)
int C(int i) {
 if (i<=5) {
 cout<<i<<" ";
return C(i+1);
 }
}

d)
void D(int i) {
 if (i>=-4) D(i-1);
 cout<<i<<" ";
}

7. Considerăm următorul program:

```
var x:byte;
function Joc( x,y:byte):byte;
begin
  write(x, ' ');
  if (x<=3) then begin
    write(y, ' ');
    inc(y);
    Joc:=Joc(x+1,y)+1;
  end
  else Joc:=0;
end;

begin
  x:=1;
  write(Joc(2,x));
end.
```

```
#include <iostream.h>
int x;
int Joc(int x, int y)
{
  cout<<x<<" ";
  if (x<=3) {
    cout<<y<<" "; y++;
    return Joc(x+1,y)+1;
  }
  else return 0;
}

void main()
{ x=1;
  cout<<Joc(2,x);
}
```

Ce valori vor fi afișate după executarea acestuia?

a) 2 1 3 2 4 2; b) 2 1 3 1 4 2; c) 2 1 3 2 4; d) 1 1 1 2 1 2.

8. Considerăm următorul program:

```
var x:byte;
function Joc(var x:byte;
              y:byte):byte;
begin
  inc(y);
  write(y, ' ');
  if (y<=5) then begin
    dec(x);
    write(x, ' ');
    Joc:=Joc(x, y+2)-1 end
  else begin
    write(x, ' ');
    Joc:=5 end
  end;
begin
  x:=3;
  writeln(Joc(x, x), ' ', x);
end.
```

```
#include <iostream.h>
int x;

int Joc(int &x, int y) {
  y++;
  cout<<y<<" ";
  if (y<=5) {
    x--;
    cout<<x<<" ";
    return Joc(x, y+2)-1;
  }
  else {
    cout<<x<<" ";
    return 5;
  }
}

void main() {
  x=3; cout<< Joc(x, x);
  cout<<" "<<x;
}
```

Ce valori se vor afișa după executarea acestuia?

a) 4 3 6 6 4 6;
 b) 4 2 7 2 4 2;
 c) 4 3 6 6 4;
 d) 4 2 7 2 4 3.

9. Considerăm următorul program:

```
var x:byte;

procedure Joc(var x:byte);
var y:byte;
begin
  if x>=8 then begin
    y:=x; dec(x);
    Joc(x);
    write(y, ' ')
  end;
  write(x, ' ')
end;

begin
  x:=10; Joc(x);
end.
```

```
#include <iostream.h>
int x;

void Joc(int &x)
{ int y;
  if (x>=8){
    y=x; x--;
    Joc(x);
    cout<<y<<" ";
  }
  cout<<x<<" ";
}

void main ()
{ x=10; Joc(x);
}
```

Ce valori se vor afișa după executarea acestuia?

a) 8 9 10; b) 7 8 7 9 8 10 9; c) 7 8 7 8 7 8 7; d) 7 8 7 9 7 10 7

10. Considerăm următorul program:

```
var x,y:byte;
procedure Joc(var x,y:byte);
begin
  write(x, ' ');
  if x>=8 then begin
    dec(x); inc(y);
    Joc(x,y);
    write(y, ' ')
  end;
end;

begin
  x:=10; y:=1;
  Joc(x,y);
end.
```

```
#include <iostream.h>
int x,y;
void Joc(int &x, int &y)
{
  cout<<x<<" ";
  if (x>=8) {
    x--; y++;
    Joc(x,y);
    cout<<y<<" ";
  }
}

void main() {
  x=10; y=1;
  Joc(x, y);
}
```

Ce valori se vor afișa după executarea acestuia?

a) 10 9 8 3 3; b) 10 9 8 3 2; c) 10 9 8 7 4 4 4; d) 10 9 8 7 4 3 2.

11. Se consideră următoarea funcție recursivă:

```
function F(x:integer):byte;
begin
  if x=0 then F:=0
  else
    if x mod 3=0 then
      F:=F(x div 10)+1
    else F:=F(x div 10)
  end;
end;
```

```
int F(int x)
{
  if (x==0) return 0;
  else
    if (x%3==0)
      return F(x/10) + 1;
    else F(x/10);
}
```

Pentru ce valoare a parametrului x , funcția F va returna valoarea 4?

a) 13369; b) 21369; c) 4; d) 1233.

12. Se consideră următoarea funcție recursivă:

```
function P(n:integer):integer;
begin
  if n<50 then
    if n mod 3=0 then
      P:=P(2*n - 3)
    else P:=P(2*n - 1)
    else P:=n;
  end;
end;
```

```
int P(int n)
{
  if (n<50)
    if (n%3==0)
      return P(2*n - 3);
    else return P(2*n - 1);
  else return n;
}
```

Pentru care dintre valorile parametrului n , funcția P va returna valoarea 61?

a) 16; b) 61; c) 4; d) 31.

13. Se consideră următorul subprogram recursiv:

```
function F(n,x:integer):byte;
begin
  if x<2 then F:=1
  else
    if n mod x=0 then F:=0
    else F:=F(n,x-1)
  end;
```

```
int F(int n,int x)
{
  if (x<2) return 1;
  else
    if (n%x==0) return 0;
    else return F(n,x-1);
}
```

Dacă $n>1$, în urma apelului $F(n, n \text{ div } 2)$ în Pascal, respectiv $F(n, n/2)$ în C++, funcția returnează valoarea 1 dacă și numai dacă:

- a) valoarea lui n reprezintă un număr care nu se divide cu $[n/2]$ (partea întreagă);
- b) valoarea lui n reprezintă un număr prim;
- c) valoarea lui n nu reprezintă un număr prim;
- d) valoarea lui n reprezintă un număr par;

14. Se consideră următoarea funcție recursivă:

```
function F(n,x:integer):byte;
begin
  if n<2 then F:=1
  else
    if n mod x=0 then
      F:=F(n div x, x+1)
    else F:=0;
  end;
```

```
int F(int n,int x)
{
  if (n<2) return 1;
  else
    if (n%x==0)
      return F(n/x, x+1);
    else return 0;
}
```

Pentru care dintre următoarele apeluri, funcția F va returna valoarea 1:

a) $F(25,2)$ b) $F(36,2)$ c) $F(24,2)$ d) $F(5040,2)$

15. Considerăm următorul subprogram recursiv:

```
procedure S(i,j:byte);
begin
  if i<=4 then
    if j<=i then begin
      write('8');
      S(i,j+1)
    end
  else begin
    writeln;
    S(i+1,1);
  end;
end;
```

```
void S(int i,int j)
{
  if (i<=4)
    if (j<=i) {
      cout<<'8';
      S(i,j+1);
    }
  else {
    cout<<endl;
    S(i+1,1);
  }
}
```

Câte caractere vor fi afișate în urma apelului $S(1,1)$?

a) 4; b) 16; c) 10; d) 8.

16. Considerăm următorul subprogram recursiv:

```
procedure S(i,j:byte);
begin
  if i<=4 then
    if j<=4 then begin
      write('6');
      S(i,j+1)
    end
    else begin
      writeln;
      S(i+1,i+2);
    end;
end;

void S(int i,int j)
{
  if (i<=4)
    if (j<=4) {
      cout<<'6';
      S(i,j+1);
    }
    else {
      cout<<endl;
      S(i+1,i+2);
    }
}
```

Câte caractere vor fi afișate în urma apelului $S(1,1)$?

- a) 4; b) 6; c) 8; d) 7.

17. Care dintre următoarele subprograme recursive returnează, în mod corect, suma cifrelor unei valori naturale primite prin parametrul x ?

<pre>a) function A(x:integer):integer; begin A:=A(x div 10)+ x mod 10; end;</pre>	<pre>a) int A(int x) { return A(x / 10)+ x % 10; }</pre>
<pre>b) function A(x:integer):integer; begin if x>9 then A:=x mod 10 + A(x div 10) end;</pre>	<pre>b) int A(int x) { if (x<9) return x % 10 + A(x / 10); }</pre>
<pre>c) function A(x:integer):integer; begin if x<>0 then A:=x mod 10 + A(x div 10) else A:=0; end;</pre>	<pre>c) int A(int x) { if (x!=0) return x % 10 + A(x / 10); else return 0; }</pre>
<pre>d) function A(x:integer):integer; begin if x<9 then A:=x mod 10 + A(x div 10) else A:=x; end;</pre>	<pre>d) int A(int x) { if (x<9) return x % 10 + A(x / 10); else return x; }</pre>

18. Considerăm următorul program:

<pre>var x:integer; function A(x:integer; var y:integer):integer; begin if x=0 then A:=y else begin y:=y * 10 + x mod 10; A:=A(x div 10, y) end end;</pre>	<pre>#include <iostream.h> int x; int A(int x, int &y) { if (x==0) return y; else { y=y*10 + x%10; return A(x / 10, y); } } void main(){ x=0; cout<<A(12031,x); }</pre>
--	--

Care dintre următoarele afirmații sunt adevărate?

- a) Programul conține o eroare de sintaxă.
 b) Programul afișează valoarea 0 deoarece 12031 nu este palindrom.
 c) Programul afișează valoarea 13021.
 d) Subprogramul A returnează oglinditul numărului primit prin parametrul x .

19. Se consideră următorul subprogram recursiv, definit incomplet:

<pre>function Min(x:integer):byte; begin if x=0 then Min:=... else if Min(x div 10)<x mod 10 then Min:=Min(x div 10) else Min:=x mod 10; end;</pre>	<pre>int Min(int x) { if (x==0) return ...; else if (Min(x / 10)<x % 10) return Min(x / 10); else return x % 10; }</pre>
--	---

Cu ce valoare trebuie înlocuite punctele de suspensie, pentru ca subprogramul să returneze cifra minimă a numărului primit prin parametrul x ?

- a) 0 b) 1 c) -1 d) 9

20. Se consideră următoarea funcție recursivă, definită incomplet:

<pre>function F(x,d:longint):integer; begin if ... then F:=d else F:=F(x,d-1) end;</pre>	<pre>int F(long x, int d) { if ... return d; else return F(x,d-1); }</pre>
--	--

Cu ce expresie trebuie înlocuite punctele de suspensie astfel încât, în urma apelului $F(x, 99)$, funcția să returneze cel mai mare divizor de cel mult două cifre al numărului natural transmis prin parametrul x ?

- | | |
|--|--|
| a) $x \bmod d < 100$ | a) $(x \% d < 100)$ |
| b) $x \bmod d = 1$ | b) $(x \% d == 1)$ |
| c) $x \bmod d = 0$ | c) $(x \% d == 0)$ |
| d) $(x \div d = 0) \text{ and } (d < 100)$ | d) $(x / 10 == 0 \text{ \&\& } d < 100)$ |

21. Se consideră următorul subprogram recursiv, definit incomplet:

<pre> procedure S(x:integer); begin write('*'); if ... then begin write('*'); S(x-1) end; end; </pre>	<pre> void S(int x) { cout<<'*'; if ... { cout<<'*'; S(x-1); } } </pre>
---	---

Cu ce expresie trebuie înlocuite punctele de suspensie astfel încât, în urma apelului $S(5)$, să se afișeze 9 de caractere '*'?

- a) $(x > 1)$ b) $(x > 0)$ c) $(x > 2)$ d) nici una din variantele anterioare

22. Considerăm următoarea funcție recursivă:

<pre> function S(n: integer):longint; begin if n=0 then S:=0 else if n mod 2=0 then S:=S(n div 10)- 2*n else S:=S(n div 10) + 3*n; end; </pre>	<pre> int S(int n) { if (n) if (n%2) return S(n/10)+3*n; else return S(n/10)-2*n; else return 0; } </pre>
--	---

Care dintre următoarele expresii au valoarea 830?

- a) $S(255)$ b) $S(253)$ c) $S(255) - 6$ d) $S(410)$

23. Considerăm următorul subprogram recursiv:

<pre> procedure S(i,j:byte); begin if i>0 then if j>0 then begin write('*'); S(i,j-1) end end; end; </pre>	<pre> void S(int i,int j) { if (i>0) if (j>0) { cout<<'*'; S(i,j-1); } } </pre>
--	---

<pre> else begin writeln; S(i-1,i-1); end; end; </pre>	<pre> else { cout<<endl; S(i-1,i-1); } } </pre>
--	---

Ce apel trebuie utilizat pentru ca pe ecran să se afișeze:

**
*

- a) $S(3, 2)$ b) $S(3, 4)$ c) $S(3, 3)$ d) $S(4, 3)$

24. Se consideră următorul subprogram:

<pre> procedure S(x,b:integer); begin if x<b then write(x) else begin S(x div b,b); write(x mod b); end; end; </pre>	<pre> void S(int x,int b) { if (x<b) cout<<x; else { S(x/b,b); cout<<x % b; } } </pre>
---	---

Ce se afișează pe ecran în urma apelului $S(305, 4)$?

- a) 10301 b) 103001 c) 131 d) 1220

25. Considerăm următorul program:

<pre> var x:byte; procedure S(var x:byte;b:byte); begin if b>0 then begin x:=x + b mod 10 + b div 10; S(x, b div 10); end end; begin x:=15; S(x,23); write(x,' '); S(x,23); write(x); end. </pre>	<pre> #include <iostream.h> int x; void S(int, &x, int b){ if (b>0){ x=x + b % 10 + b / 10; S(x, b / 10); } } void main(){ x=15; S(x,23); cout<<x<<" "; S(x,23); cout<<x; } </pre>
---	--

Ce se va afișa în urma rulării acestuia:

- a) 22 22 b) 22 29 c) 15 15 d) 29 29

26. Considerăm următorul program:

```
var x:byte;
function F(var x:byte;b:byte):byte;
begin
  if b<>0 then begin
    x:=x + b mod 10;
    F:=F(x, b div 10)+ x;
  end
  else F:=0;
end;

begin
  x:=0;
  write(F(x,123),' ');
  write(F(x,123));
end;
```

```
#include <iostream.h>
int x;
int F(int &x,int b){
  if (b!=0){
    x=x + b % 10;
    return F(x, b / 10)+ x;
  }
  else return 0;
}

void main(){
  x=0;
  cout<<F(x,123)<<' ';
  cout<<F(x,123);
}
```

Ce se va afișa în urma rulării acestuia:

- a) 14 14 b) 18 18 c) 0 0 d) 18 36

2.2.2 Probleme rezolvate

1. Considerăm un șir de n valori naturale ($n \leq 50$), reținute în tabloul unidimensional a . Realizați un subprogram care permite afișarea acestora în ordinea a_n, a_{n-1}, \dots, a_1 . Implementați două variante recursive ale acestui subprogram. Specificați modul în care poate fi apelat subprogramul din programul principal (în Pascal), respectiv din cadrul funcției *main* (în C++). Vectorul și numărul de elemente ale acestuia le considerăm definite ca variabile globale.

Exemplu: Pentru $n=5$ și șirul 2, 4, 5, 7, 8 subprogramul va afișa pe ieșirea standard valorile 8, 7, 5, 4, 2.

Soluție:

Prima variantă a subprogramului implementează recursiv instrucțiunea *For* al cărui contor își decrementează valoarea la fiecare iterație. Practic, este vorba de pacurgerea în ordine inversă a elementelor vectorului.

A doua variantă a subprogramului, folosește proprietatea recursivității de revenire în subprograme pentru executarea instrucțiunilor abandonate, în ordinea inversă a autoapelurilor (se respectă mecanismul stivei *Last in-Firs out*).

Ambele subprograme au un singur parametru întreg, reprezentând indicele elementului curent din vector.

Subprogramul va fi implementat în Pascal ca procedură, iar în C++ ca funcție de tip *void*.

```
1  VARIANTA 1
2  procedure Afis(i:byte);
3  begin
4    if i>0 then begin
5      write(a[i]);
6      Afis(i-1)
7    end
8  end;
9
10 begin
11   ... Afis(n); ...
12 end.

13  VARIANTA 2
14  procedure Afis(i:byte);
15  begin
16    if i<=n then begin
17      Afis(i+1);
18      write(a[i])
19    end
20  end;
21
22 begin
23   ... Afis(1); ...
24 end.
```

```
VARIANTA 1
void Afis(int i)
{
  if (i>=0) {
    cout<<a[i];
    Afis(i-1);
  }
}

void main(){
  ... Afis(n-1); ...
}

VARIANTA 2
void Afis(int i)
{
  if (i<n){
    Afis(i+1);
    cout<<a[i];
  }
}

void main(){
  ... Afis(0); ...
}
```

2. Realizați un subprogram recursiv care să permită memorarea, într-un vector, a primelor n ($n \leq 50$) numere pare în ordine descrescătoare. Subprogramul va fi implementat în manieră recursivă. Considerăm că n este o variabilă declarată global. Specificați modul în care poate fi apelat subprogramul din programul principal (în Pascal), respectiv din cadrul funcției *main* (în C++).
Exemplu: Pentru $n=5$ subprogramul va construi un vector ale cărui elemente vor fi 8, 6, 4, 2, 0.

Soluție:

Subprogramul va avea doi parametri: un parametru referință reprezentând vectorul ale cărui elemente vor fi memorate în cadrul subprogramului și un parametru valoare întreg ce va reprezenta indicele elementului curent. Subprogramul va fi implementat în Pascal ca procedură, iar în C++ ca funcție de tip *void*.

```
1  procedure C(i:byte;var a:sir);
2  begin
3    if i<=n then begin
4      a[i]:=2*(n-i);
5      C(i+1,a)
6    end
7  end;
8
9  begin
10   ... C(1,a); ...
11 end.
```

```
void C(int i, int a[])
{
  if (i<n) {
    a[i]=2*(n-i-1);
    C(i+1,a);
  }
}

void main(){
  ... C(0,a); ...
}
```

3. Considerăm un șir de n valori naturale ($n \leq 50$), reținute în tabloul unidimensional a . Implementați două variante recursive ale unui subprogram care determină elementul minim al tabloului. Considerăm definite global atât variabila n , cât și vectorul a . În varianta Pascal, subprogramul va fi implementat într-o variantă ca funcție, iar în cea de a doua ca procedură. În varianta C++, una din funcții va fi de tip *int*, iar cea de a doua de tip *void*. Specificați modul în care poate fi apelat subprogramul, din programul principal (în Pascal), respectiv din cadrul funcției *main* (în C++), pentru a fi afișat elementul minim determinat.

Exemplu: Pentru $n=5$ și șirul 12, 4, 5, 7, 8, subprogramul va determina minimul 4.

Soluție:

În cazul primei variante de implementare, funcția va returna un rezultat întreg reprezentând elementul minim determinat. Parametrul întreg i indică poziția elementului curent din vector.

În cazul celei de a doua variante de implementare, rezultatul va fi transmis prin intermediul parametrului referință x . Valoarea acestuia se va actualiza la revenirea din recursivitate.

```

1  VARIANTA 1
2  function M(i:byte):word;
3  begin
4    if i=n then M :=a[n]
5  else
6    if a[i]<M(i+1) then
7      M :=a[i]
8    else M :=M(i+1)
9  end;
10
11 begin
12   ... write(M(1)); ...
13 end.

14  VARIANTA 2
15 procedure M(i:byte;var x:word);
16 begin
17   if i=n then x:=a[n]
18 else begin
19   M(i+1,x);
20   if a[i]<x then x:=a[i];
21 end
22 end;
23
24 begin
25   ...M(1,x); writeln(x, ' ');...
26 end.
```

```

1  VARIANTA 1
2  int M(int i)
3  {
4    if (i==n-1) return a[n-1];
5  else
6    if (a[i]<M(i+1))
7      return a[i];
8    else return M(i+1);
9  }

10 void main(){
11   ... cout<<M(0); ...
12 }

13  VARIANTA 2
14 void M(int i,int &x)
15 {
16   if (i==n-1) x=a[n-1];
17 else{
18   M(i+1,x);
19   if (a[i]<x) x=a[i];
20 }
21 }

22 void main(){
23   ... M(0,x);cout<<x;...
24 }

```

4. Se consideră două tablouri unidimensionale: a de lungime n și b de lungime m , ale căror elemente naturale sunt ordonate crescător. Să se realizeze subprogramul recursiv *Inter* care realizează operația de *interclasare* a elementelor celor două tablouri în vectorul c . Subprogramul va avea trei parametri valoare i, j, k . Primii doi

reprezintă, în ordine, pozițiile elementelor din a respectiv din b , care se compară la pasul curent. Parametrul k indică poziția pe care este plasat elementul minim în vectorul c . Considerăm că cele trei tablouri și lungimile lor sunt declarate global.

Exemplu: Pentru $n=5, m=3$ și tablourile $a=(1, 4, 5, 7, 8)$, $b=(2,17,18)$, subprogramul *Inter* plasează elementele în vectorul c în ordinea (1,2,4,5,7,8,17,18).

Soluție:

Algoritmul de interclasare parcurge elementele celor doi vectori realizând compararea succesivă a elementelor curente. Compararea începe cu elementele situate pe prima poziție, cel mai mic fiind plasat într-un nou vector (C). Se înaintează cu o poziție în vectorul din care s-a copiat elementul plasat în C , ș.a.m.d. Subprogramul va avea trei parametri valoare i, j, k , reprezentând, în ordine, pozițiile elementelor din a și b care se compară la pasul curent și, respectiv, poziția pe care este plasat elementul minim în vectorul c . Condiția de oprire a recursivității este $(i>n)$ and $(j>m)$ – Pascal, respectiv $(i>n \ \&\& \ j>m)$ în C++. Subprogramul va fi implementat în Pascal ca procedură, iar în C++ ca funcție de tip *void*.

<pre> 1 procedure Inter(i,j,k:byte); 2 begin 3 if (i<=n)and(j<=m) then 4 if a[i]<b[j] then begin 5 c[k]:=a[i]; 6 Inter(i+1,j,k+1);end 7 else begin 8 c[k]:=b[j]; 9 Inter(i,j+1,k+1); 10 end 11 else 12 if i<=n then begin 13 c[k]:=a[i]; 14 Inter(i+1,j,k+1); end 15 else 16 if j<=m then begin 17 c[k]:=b[j]; 18 Inter(i,j+1,k+1); end 19 end;</pre>	<pre> 1 void Inter(int i,int j,int k) 2 { 3 if (i<n && j<m) 4 if (a[i]<b[j]) { 5 c[k]=a[i]; 6 Inter(i+1,j,k+1); 7 } 8 else { 9 c[k]=b[j]; 10 Inter(i,j+1,k+1); 11 } 12 else 13 if (i<n){ 14 c[k]=a[i]; 15 Inter(i+1,j,k+1); 16 } 17 else 18 if (j<m){ 19 c[k]=b[j]; 20 Inter(i,j+1,k+1); 21 } 22 }</pre>
---	--

5. Considerăm un șir de n valori naturale ($n \leq 50$), reținute în tabloul unidimensional a . Să se realizeze subprogramul recursiv *Det* care creează un vector b cu elementele distincte din a . Funcția recursivă *Ok* va fi apelată din subprogramul *Det* pentru a verifica dacă o valoare transmisă ca parametru se găsește în vectorul b . Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++. Vectorii și variabila n sunt considerate globale.

Subprogramul *Det* va avea doi parametri întregi i și k reprezentând, în ordine, poziția elementului curent din vectorul a respectiv lungimea vectorului b .

Funcția *Ok* va avea trei parametri întregi: x , reprezentând valoarea căutată în vectorul b , i , reprezentând indicele curent din vectorul b și k , reprezentând lungimea acestuia. Scrieți definițiile complete ale două subprograme.

Exemplu: Pentru $n=5$ și vectorul $a=(1, 8, 1, 7, 8)$ subprogramul *Det* va permite memorarea în vectorul b a valorilor 1, 8, 7.

Soluție:

Funcția *Ok* parcurge elementele vectorului b , oprirea recursivității fiind posibilă în situația în care valoarea lui x este egală cu elementul curent din b , sau în cazul în care nici un element nu a fost egal cu x , deci $i > k$.

Subprogramul *Det* returnează, prin parametrul referință k , numărul de valori distincte din vectorul a . Parametrul valoare i reprezintă poziția elementul curent din vectorul a . Subprogramul va fi implementat în Pascal ca procedură, iar în C++ ca funcție de tip *void*.

<pre> 1 function Ok(x,i,k:byte): 2 boolean; 3 begin 4 if i>k then Ok:=false 5 else 6 if x=b[i] then Ok:=true 7 else Ok:=Ok(x,i+1,k); 8 end; 9 10 procedure Det(i:byte; 11 var k:byte); 12 begin 13 if (i<=n) then begin 14 if not Ok(a[i],1,k) then 15 begin 16 inc(k); 17 b[k]:=a[i]; 18 end; 19 Det(i+1,k); 20 end; 21 end;</pre>	<pre> int Ok(int x,int i,int k) { if (i>=k) return 0; else if (x==b[i]) return 1; else return Ok(x,i+1,k); } void Det(int i,int &k) { if (i<n) { if (!Ok(a[i],0,k)){ k++; b[k-1]=a[i]; } Det(i+1,k); } }</pre>
---	---

6. Considerăm un șir de n valori naturale ($n \leq 50$), reținute în tabloul unidimensional a . Realizați subprogramul recursiv *Nr* care determină numărul de elemente prime din vector. Rezultatul va fi returnat prin intermediul unui parametru întreg. Subprogramul va apela funcția recursivă *Ok*, care verifică dacă o valoare primită ca parametru reprezintă un număr prim. Funcția returnează, în Pascal, valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++. Vectorul a și lungimea n a acestuia sunt considerate variabile globale.

Scrieți definițiile complete ale celor două subprograme.

Exemplu: Pentru $n=5$ și vectorul $a=(11, 8, 13, 7, 8)$ subprogramul *Nr* va returna valoarea 3.

Soluție:

Funcția *Ok* va avea doi parametri întregi x și i . Parametrul x reprezintă valoarea care se verifică dacă este un număr prim, iar i reprezintă un posibil divizor al lui. Aceasta din urmă își mărește valoarea cu o unitate la fiecare autoapel. Numărul

este prim dacă valoarea lui i ajunge în urma autoapelurilor mai mare decât $\lfloor x/2 \rfloor$. Se poate limita valoarea maximă a lui i la $\lfloor \sqrt{x} \rfloor$.

Subprogramul *Nr* returnează, prin parametrul referință k , numărul de numere prime din vectorul a . Parametrul valoare i reprezintă poziția elementul curent din vector. Subprogramul *Nr* va fi implementat în Pascal ca procedură, iar în C++ ca funcție de tip *void*.

<pre> 1 function Ok(x,i:integer): 2 boolean; 3 begin 4 if i>x div 2 then Ok:=true 5 else 6 if x mod i=0 then Ok:=false 7 else Ok:=Ok(x,i+1); 8 end; 9 10 procedure Nr(i:integer; 11 var k:byte); 12 begin 13 if (i<=n) then begin 14 if Ok(a[i],2) then inc(k); 15 Nr(i+1,k); 16 end; 17 end;</pre>	<pre> int Ok(int x, int i) { if (i>x/2) return 1; else if (x%i==0) return 0; else return Ok(x,i+1); } void Nr(int i,int &k) { if (i<n) { if (Ok(a[i],2)) k++; Nr(i+1,k); } }</pre>
---	---

7. Considerăm un șir de n valori naturale ($n \leq 50$), reținute în tabloul unidimensional a . Realizați subprogramul recursiv *Cm*, care determină cifra maximă ce apare în scrierea în baza 10 a tuturor elementelor vectorului a și numărul de apariții ale acesteia. Ambele rezultate vor fi returnate prin intermediul a doi parametri. În cadrul subprogramului se va face apel la alte două subprograme recursive:

- funcția *Mx*, care determină cifra maximă a unei valori transmise ca parametru;
- funcția *Nr*, care primește două valori întregi prin parametri x și c . Aceasta returnează numărul de apariții al cifrei c în numărul x .

Scrieți definițiile complete ale celor trei subprograme. Vectorul a și lungimea n a acestuia sunt considerate variabile globale.

Exemplu: Pentru $n=5$ și vectorul $a=(121, 38, 183, 7, 8)$, subprogramul *Cm* va returna, prin cei doi parametri, cifra 8 și valoarea 3 reprezentând numărul ei de apariții.

Soluție:

Subprogramul *Cm* returnează prin parametrii c și m , cifra maximă a tuturor elementelor vectorului a și numărul de apariții ale acesteia. Parametrul i reprezintă poziția elementului curent din vector. Valoarea lui c se actualizează, după caz, cu cea mai mare cifra a elementului $a[i]$, adică valoarea returnată la apelul *Mx*($a[i]$). De asemenea, valoarea lui m se mărește sau se reinițializează cu valoarea returnată de funcția *Nr* în urma apelului *Nr*($a[i],c$).

Subprogramul *Cm* va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip *void*.

```

1 function Nr(x,c:integer):
2     integer;
3 begin
4     if x=0 then Nr:=0
5     else
6         if x mod 10=c then
7             Nr:=Nr(x div 10, c)+1
8         else Nr:=Nr(x div 10, c);
9 end;
10
11 function Mx(x:integer):byte;
12 begin
13     if x=0 then Mx:=0
14     else
15         if x mod 10>Mx(x div 10)
16         then Mx:=x mod 10
17         else Mx:=Mx(x div 10)
18 end;
19
20 procedure Cm(i:integer;
21     var c,m:integer);
22 begin
23     if i<=n then begin
24         if Mx(a[i])>c then begin
25             c:=Mx(a[i]);
26             m:=Nr(a[i],c);
27         end
28         else
29             if Mx(a[i])=c then
30                 m:=m + Nr(a[i],c);
31             Cm(i+1,c,m);
32         end;
33 end;

```

8. Realizați un subprogram recursiv care să permită citirea de la tastatură a unui șir de *n* valori întregi și să le afișeze pe cele care conțin un număr maxim de cifre distincte. Subprogramul nu va utiliza date structurate. El va apela funcția recursivă *D*, care returnează numărul de cifre distincte ale unei valori întregi primite ca parametru. La rândul său, funcția *D* va apela funcția recursivă *Nr*, care determină numărul de apariții al unei cifre într-un număr. Ambele valori vor fi transmise prin parametri. Scrieți definițiile complete ale celor trei subprograme, considerând variabila *n* definită global.

Exemplu: Pentru *n*=5 și valorile 121, 38, 7, 1188, 22 se va afișa 1188, 38, 121

Soluție:

Funcția *D* determină numărul de cifre distincte ale parametrului *x* prin numărarea cifrelor de la 0 la 9 care apar cel puțin o dată în scrierea acestuia. Cele 10 cifre sunt

reținute prin intermediul parametrului valoare *i*. În acest fel, la apelul *D(x,0)* se returnează numărul de cifre distincte ale lui *x*.

Subprogramul *P* permite citirea fiecărei valori din șirul celor *n*. Numărul de cifre distincte ale fiecărei valori citite va fi salvat cu ajutorul variabilei locale *y*. La revenirea din recursivitate vor fi afișate acele valori pentru care valoarea variabilei *y* este egală cu valoarea finală a lui *max*.

Subprogramul *P* va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip *void*.

```

1 var n,max:integer;
2
3 function Nr(x,c:integer):
4     integer;
5 begin
6     if x=0 then Nr:=0
7     else
8         if x mod 10=c then
9             Nr:=Nr(x div 10, c) + 1
10        else Nr:=Nr(x div 10, c);
11 end;
12
13 function D(x,i:integer):byte;
14 begin
15     if i>9 then D:=0
16     else
17         if Nr(x,i)>=1 then
18             D:=D(x,i+1)+1
19         else D:=D(x,i+1)
20 end;
21
22 procedure P(i:integer;
23     var max:integer);
24 var x,y:integer;
25 begin
26     if i<=n then begin
27         readln(x); y:=D(x,0);
28         if y>max then max:=y;
29         P(i+1,max);
30         if max=y then write(x,' ');
31     end;
32 end;
33
34 begin
35     max:=0; readln(n); P(1,max);
36 end.

```

9. Realizați un subprogram recursiv *Nr* care permite citirea a *n* șiruri de caractere de lungimi egale și care returnează, prin intermediul unui parametru, numărul de șiruri ce sunt anagrame cu ultimul citit. Subprogramul nu va utiliza tablouri pentru memorarea acestora.

În cadrul lui se va apela funcția *Ok*, care verifică dacă două șiruri de caractere

primate prin intermediul a doi parametri sunt anagrame. Funcția *Ok* returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++. Considerăm *n* variabilă globală.

Exemplu: Pentru $n=4$ și șirurile *arc*, *rac*, *dar*, *car* subprogramul *Nr* va returna printr-un parametru întreg valoarea 2.

Soluție:

Funcția *Ok* caută succesiv primul caracter al parametrului *a*, în cel de al doilea (*b*). În cazul în care este găsit va fi șters și din *a* și din *b*, funcția autoapelându-se pentru noile valori ale parametrilor. Procedeu continuă fie până când caracterul nu este găsit, caz în care funcția returnează valoarea *False*/0, fie când lungimea lui *a* este 0, caz în care cuvintele sunt anagrame.

Subprogramul *Nr* va avea doi parametri referință:

- parametrul întreg *m*, care contorizează numărul „cuvinte” anagrame cu ultimul șir citit;
- parametrul șir de caractere *y*, care va reține ultimul șir de caractere citit.

La revenirea din autoapeluri, parametrul *m* își mărește valoarea cu o unitate, dacă la apelul *Ok(x,y)*, se returnează valoarea *True*/1. Parametrul efectiv *x* va transmite, la fiecare apel, șirul de caractere citit la pasul curent. Subprogramul *Nr* va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip *void*.

```

1 function Ok(a,b:string): boolean;
2
3 begin
4   if a='' then Ok:=true
5   else
6     if pos(a[1],b)<>0 then
7       begin
8         delete(b,pos(a[1],b),1);
9         delete(a,1,1);
10        Ok:=Ok(a,b);
11      end
12     else Ok:=false;
13   end;
14
15 procedure Nr(i:byte;
16   var m:byte; var y:string);
17 var x:string;
18 begin
19   if i=n then readln(y)
20   else begin
21     readln(x);
22     Nr(i+1,m,y);
23     if Ok(x,y) then inc(m);
24   end;
25 end;
26
int Ok(char a[], char b[])
{
  int p;
  char c[256];
  strcpy(c,b);
  if (strlen(a)==0) return 1;
  p=strchr(c,a[0])-c;
  if (p<strlen(c)) {
    strcpy(c+p,c+p+1);
    strcpy(a,a+1);
    return Ok(a,c);
  }
  else return 0;
}

void Nr(int i,int &m,
  char y[])
{
  char x[256];
  if (i==n) cin>>y;
  else {
    cin>>x;
    Nr(i+1,m,y);
    if (Ok(x,y)) m++;
  }
}

```

10. Considerăm un șir de *n* valori naturale ($n \leq 50$), reținute într-un tablou unidimensional. Realizați subprogramul recursiv *M*, care primește, prin parametrul *a*, un astfel de vector și care înlocuiește fiecare element *a[i]* cu suma cifrelor lui care au aceeași paritate cu indicele *i*.

Subprogramul va apela funcția *C*, care calculează la alegere suma cifrelor pare sau impare ale unei valori transmise prin parametrul întreg *x*. Parametrul *t* al funcției va primi la apel valoarea 1 dacă se dorește calcularea sumei cifrelor impare și 0 în caz contrar. Considerăm *n*, variabilă globală.

Exemplu: Pentru $n=5$ și valorile 121, 382, 7, 1188, 22, se va afișa, în Pascal, 2, 10, 7, 16, 0 iar în C++, 2, 3, 0, 2, 4

Soluție:

Funcția *C* parcurge cifră cu cifră numărul transmis prin parametrul *x* verificând dacă restul la împărțirii la 2 este egal cu *t*.

Subprogramul *M* are, ca parametru referință, vectorul *a* și, ca parametru valoare, indicele *i*. La fiecare autoapel se actualizează elementul *a[i]* cu valoarea returnată în urma apelului *C(a[i],i mod 2)* (în Pascal), respectiv *C(a[i],i%2)* (în C++).

Subprogramul *M* va fi implementat în Pascal ca procedură, iar în C++ ca funcție de tip *void*. În Pascal considerăm definit tipul de date *sir=array[1..50] of integer*.

```

1 function C(x,t:integer):byte;
2 begin
3   if x=0 then C:=0
4   else
5     if x mod 2=t then
6       C:=C(x div 10,t)+x mod 10
7     else C:=C(x div 10,t)
8   end;
9
10 procedure M(i:byte;var a:sir);
11 begin
12   if i<=n then begin
13     a[i]:=C(a[i],i mod 2);
14     M(i+1,a)
15   end
16 end;

int C(int x,int t)
{
  if (x==0) return 0;
  else
    if (x % 2==t)
      return C(x/10,t) + x%10;
    else return C(x/10,t);
}

void M(int i,int a[])
{
  if (i<n){
    a[i]=C(a[i],i%2);
    M(i+1,a);
  }
}

```

11. Un elev citește un text ce conține *n* cuvinte, pe care îl codifică după următoarea regulă: inversează cuvintele în cadrul textului și literele în cadrul cuvintelor. Realizați un subprogram recursiv *M*, care permite citirea celor *n* cuvinte și care afișează textul obținut în urma procesului de codificare. Subprogramul nu va conține tablouri pentru memorarea cuvintelor. Acesta va face apel la funcția recursivă *V*, care returnează inversul unui șir de caractere primit prin intermediul unui parametru. Considerăm *n* variabilă globală.

Exemplu: Pentru $n=4$ și cuvintele: *ana are multe mere*, subprogramul va afișa: *eremetlumeraana*

Soluție:

Funcția V determină inversul unui cuvânt prin concatenare în ordine inversă a literelor acestuia. Astfel, prima literă a parametrului s va fi concatenat la finalul inversului cuvântului care începe de la a doua literă.

Subprogramul M are un singur parametru întreg i , care indică numărul de ordine al cuvântului citit. Acesta va fi memorat în variabila locală s . Cuvântul returnat în urma apelului $V(s)$ va fi afișat la revenirea din recursivitate.

Subprogramul M va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip `void`.

```
1 function V(s:string):string;
2 begin
3   if s='' then V:=''
4   else
5     V:=V(copy(s,2,length(s)-1))+s[1]
6   end;
7
8 procedure M(i:byte);
9 var s:string;
10 begin
11   if i<=n then begin
12     readln(s);
13     M(i+1);
14     write(V(s))
15   end
16 end;
```

```
char* V(char s[])
{
  char t[256] = {0};
  if (strlen(s)==0) return s;
  else {
    strcpy(t,V(s+1));
    t[strlen(t)]=s[0];
    return t;
  }
}

void M(int i) {
  char s[256];
  if (i<=n) { cin>>s; M(i+1);
    cout<<V(s); }
}
```

12. Considerăm un șir de n valori naturale ($n \leq 50$), reținute în tabloul unidimensional a . Realizați un subprogram recursiv S care să permită ordonarea la alegere (crescător/descrescător) a elementelor lui. Tipul ordonării care se dorește a fi realizată va fi transmisă codificat prin parametrul întreg t . Dacă, la apel, acestuia i se transmite valoarea 1, atunci se va realiza o ordonare ascendentă, iar dacă valoarea transmisă acestuia este -1, sortarea va fi descendentă.

Realizați două variante de definiție a lui S . Una dintre ele nu va conține nici o structură repetitivă. Vectorul a și lungimea n a acestuia sunt considerate variabile globale.

Exemplu: Considerăm $n=5$ și tabloul (10, 234, 21, 1, 34). În urma apelului la subprogramul S pentru $t=-1$ elementele vectorului vor fi (234, 34, 21, 10, 1).

Soluție:

În prima variantă de implementare, subprogramul S are doi parametri valoare întregi i și t . Variabila locală j reprezintă contorul unei instrucțiuni *For*, care parcurge elementele cu indici mai mari ca i .

În a doua variantă de implementare S are trei parametri valoare întregi i , j și t . Subprogramul nu conține nici o structură repetitivă. El va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip `void`. În varianta Pascal, se consideră definit tipul de date `sir=array[1..50] of integer`.

150

```
1      VARIANTA 1
2 procedure S(i,t:integer);
3 var x,j:integer;
4 begin
5   if i<n then
6     for j:=i+1 to n do
7       if a[i]*t>a[j]*t then begin
8         x:=a[i]; a[i]:=a[j];
9         a[j]:=x;
10      end
11   else S(i+1,t)
12 end;
13
14      VARIANTA 2
15 procedure S(i,j,t:integer);
16 var x:integer;
17 begin
18   if i<n then
19     if j<=n then begin
20       if a[i]*t>a[j]*t then begin
21         x:=a[i];
22         a[i]:=a[j];
23         a[j]:=x;
24       end;
25       S(i,j+1,t)
26     end
27   else S(i+1,i+2,t)
28 end;
```

```
      VARIANTA 1
void S(int i,int t)
{
  int x, j;
  if (i<n-1)
    for (j=i+1;j<n;j++)
      if (a[i]*t>a[j]*t){
        x=a[i]; a[i]=a[j];
        a[j]=x;
      }
    else S(i+1,t);
}

      VARIANTA 2
void S(int i,int j,int t)
{
  int x;
  if (i<n-1)
    if (j<n) {
      if (a[i]*t>a[j]*t){
        x=a[i];
        a[i]=a[j];
        a[j]=x;
      }
      S(i,j+1,t);
    }
  else S(i+1,i+2,t);
}
```

13. Scrieți un subprogram recursiv care afișează pe ecran n linii conținând numere naturale sub forma următoare(considerăm n variabilă globală):

Exemplu: Pentru $n=4$ se va afișa:

```
1
1 2 3
1 2 3 4 5
1 2 3 4 5 6 7
```

Soluție:

Subprogramul S are doi parametri valoare întregi i și j , reprezentând numărul de ordine al liniei pe care se face afișarea, respectiv valoarea care va fi afișată. Subprogramul nu conține nici o structură repetitivă. El va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip `void`.

```
1 procedure S(i,j:integer);
2 begin
3   if i<=n then
4     if j<=2*i-1 then begin
5       write(j,' ');
6       S(i,j+1)
7     end
8   else begin
9     writeln; S(i+1,1)
10  end;
11 end;
```

```
void S(int i,int j)
{
  if (i<=n)
    if (j<=2*i-1){
      cout<<j<<' ';
      S(i,j+1);
    }
  else {
    cout<<endl; S(i+1,1);
  }
}
```

14. Scrieți un subprogram recursiv care, primind printr-un parametru un număr natural strict pozitiv n , afișează pe ecran $2 \cdot n - 1$ linii conținând caractere sub forma următoare:

Exemplu: Pentru $n=3$ se va afișa:

```
*
**
***
**
*
```

Soluție:

Subprogramul S are trei parametri valoare n , i și j . Valoarea parametrului i reprezintă numărul de ordine al liniei pe care se face afișarea. Parametrul j indică numărul de ordine (în cadrul liniei) al caracterului care va fi afișat. Subprogramul nu conține nici o structură repetitivă. El va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip *void*.

<pre>1 procedure S(n,i,j:integer); 2 begin 3 if i<=n then 4 if j<=i then begin 5 write('* '); 6 S(n,i,j+1) 7 end 8 else begin 9 writeln; 10 S(n,i+1,1) 11 end 12 else 13 if i<=2*n then 14 if j<2*n-i+1 then begin 15 write('* '); 16 S(n,i,j+1) 17 end 18 else begin 19 writeln; 20 S(n,i+1,1) 21 end 22 end;</pre>	<pre>void S(int n, int i, int j) { if (i<=n) if (j<=i){ cout<<"* "; S(n,i,j+1); } else { cout<<endl; S(n,i+1,1); } else if (i<=2*n) if (j<2*n-i+1){ cout<<"* "; S(n,i,j+1); } else { cout<<endl; S(n,i+1,1); } }</pre>
--	--

15. Considerăm un șir de n valori naturale ($n \leq 50$), reținute în tabloul unidimensional a . Realizați un subprogram recursiv S care construiește alți doi vectori ce vor conține elementele lui a care au prima cifră pară, respectiv cele care au prima cifră impară. Subprogramul va returna vectorii construiți și lungimile acestora prin intermediul parametrilor. El va apela funcția recursivă C , care determină prima cifră a unei valori naturale primite prin intermediul unui parametru. Considerăm tabloul a și lungimea n a acestuia declarate global.

Scrieți definițiile complete ale celor două subprograme.

Exemplu: Pentru $n=5$ și $a=(10, 234, 21, 1, 34)$, subprogramul S va construi doi vectori de lungime 3, respectiv 2: $(10, 1, 34)$ și $(234, 21)$.

Soluție:

Transferul de date pentru subprogramul S se realizează prin cinci parametri: parametrul valoare i , și patru parametri referință: tablourile b și d , respectiv lungimile lor, k și l . Dacă, în urma apelului $C(a[i])$, se returnează o valoare pară, atunci elementul $a[i]$ este memorat în vectorul b , altfel în vectorul d .

Subprogramul S va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip *void*. În varianta Pascal se consideră definit tipul de date $sir=array[1..50]$ of *word*.

<pre>1 function C(x:word):byte; 2 begin 3 if x<10 then C:=x 4 else C:=C(x div 10); 5 end; 6 7 procedure S(i:byte; 8 var b,d:sir;var k,l:byte); 9 begin 10 if i<=n then begin 11 if C(a[i]) mod 2=0 then 12 begin 13 inc(k); 14 b[k]:=a[i]; 15 end 16 else begin 17 inc(l); 18 d[l]:=a[i]; 19 end; 20 S(i+1,b,d,k,l); 21 end; 22 end;</pre>	<pre>int C(int x) { if (x<10) return x; else return C(x/10); } void S(int i,int b[], int d[],int &k,int &l) { if (i<n){ if (C(a[i])%2==0) { b[k]=a[i]; k++; } else { d[l]=a[i]; l++; } S(i+1,b,d,k,l); } }</pre>
--	---

16. Considerăm două șiruri de n cifre zecimale ($n \leq 10$), memorate în tablourile unidimensionale a și b . Să se realizeze funcția recursivă E , care returnează valoarea expresiei $a[1]^{b[1]} + a[2]^{b[2]} + \dots + a[n]^{b[n]}$. În cadrul acesteia se va face apel la funcția recursivă Pow , care primind prin parametrul x și y două numere naturale, calculează valoarea x^y .

Tablourile și lungimea acestora se consideră a fi declarate ca variabile globale.

Exemplu: Pentru $n=5$ și tablourile: $a=(1,2,3,4,5)$ și $b=(5,4,3,2,1)$, se va afișa 65.

Soluție:

Funcția Pow calculează valoarea x^y prin înmulțiri repetate. Condiția de oprire a recursivității este îndeplinită când y este egal cu 0, caz în care funcția returnează valoarea 1. Funcția E primește un număr natural prin parametrul valoare i . Acesta indică poziția curentă a elementelor din cei doi vectori. În varianta Pascal se consideră definit tipul de date $sir=array[1..10]$ of *byte*.

```

1 function Pow(x,y:byte):
2     longint;
3 begin
4     if y=0 then Pow:=1
5     else
6         Pow:=x*Pow(x,y-1);
7 end;
8
9 function E(i:byte):longint;
10 begin
11     if i<=n then
12         E:=Pow(a[i],b[i])+E(i+1)
13     else
14         E:=0;
15 end;

```

```

long Pow(int x,int y)
{
    if (y==0) return 1;
    else
        return x*Pow(x,y-1);
}

long E(int i)
{
    if (i<n)return
        Pow(a[i],b[i])+E(i+1);
    else
        return 0;
}

```

17. Considerăm șirul 1, 3, 6, 10, 15, 21, 28, 36....Să se realizeze cu subprogram recursiv M care permite memorarea primilor n^2 termeni ai șirului într-o matrice pătratică cu n linii și n coloane ($n < 10$). Termenii vor fi plasați în ordine pe linii de la stînga la dreapta. Subprogramul va apela funcția recursivă T , care returnează termenul șirului cu numărul de ordine transmis prin intermediul unui parametru întreg. Variabila n se consideră declarată global. Scrieți definițiile complete ale celor două subprograme.

Exemplu: Pentru $n=4$, subprogramul M va memora în matrice elementele:

```

1 3 6 10
15 21 28 36
45 55 66 78
91 105 120 136

```

Soluție:

Termenii șirului respectă o recurență simplă. Astfel, termenul al n -lea se deduce după regula $T(n)=n+T(n-1)$ pornind de la $T(1)=1$.

Funcția recursivă T implementează această recurență.

Subprogramul M primește două valori naturale prin parametrii i și j , valori care reprezintă indicele liniei și al coloanei elementului curent. Matricea creată este returnată prin parametrul referință a . Elementul $a[i,j]$ va memora termenul din șir cu numărul de ordine pe care îl are $a[i,j]$ la liniarizarea matricei.

Subprogramul M va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip *void*. În varianta Pascal se consideră definit tipul de date $mat=array[1..10,1..10]$ of *longint*.

```

1 function T(x:byte):longint;
2 begin
3     if x=1 then T:=1
4     else T:=x+T(x-1);
5 end;

```

```

long T(int x)
{
    if (x<=1) return 1;
    else return x+T(x-1);
}

```

```

6 procedure M(i,j:byte;
7     var a:mat);
8 begin
9     if i<=n then
10         if j<=n then begin
11             a[i,j]:=T(n*(i-1)+j);
12             M(i,j+1,a);
13         end
14         else M(i+1,1,a);
15 end;

```

```

void M(int i,int j,
    long a[10][10])
{
    if (i<n)
        if (j<n) {
            a[i][j]=T(n*i+j+1);
            M(i,j+1,a);
        }
        else M(i+1,0,a);
}

```

18. Se citește de la tastatură un șir de n valori naturale de cel mult 9 cifre. Se dorește afișarea, în ordine inversă, a valorilor citite din care au fost șterse cifrele impare. Subprogramul recursiv M efectuează prelucrarea cerută, fără a folosi tablouri pentru reținerea valorilor citite. Subprogramul apelează funcția recursivă T , care elimină cifrele impare ale unei valori transmise ca parametru. Variabila n se consideră a fi declarată global. Scrieți definițiile complete ale celor două subprograme.

Exemplu: Pentru $n=5$ și valorile 1324, 4325, 4356, 492181, 32474 se va afișa:
244, 428, 46, 42, 24

Soluție:

Funcția T primește o valoare naturală prin intermediul unui parametru valoare - întreg lung. Acesta va fi parcurs cifră cu cifră, valoarea returnată de funcție reprezentând un număr creat doar din cifrele lui pare.

Subprogramul M are un singur parametru întreg i , care indică numărul de ordine al numărului citit. Acesta va fi memorat în variabila locală x . Numărul returnat în urma apelului $T(x)$ va fi afișat la revenirea din recursivitate. Subprogramul M va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip *void*.

```

1 function T(x:longint):longint;
2 begin
3     if x=0 then T:=0
4     else
5         if x mod 2=0 then
6             T:=x mod 10+10*T(x div 10)
7         else T:=T(x div 10)
8     end;
9
10 procedure M(i:byte);
11 var x:longint;
12 begin
13     if i<=n then begin
14         readln(x); M(i+1);
15         write(T(x),' ');
16     end
17 end;

```

```

long T(long x)
{
    if (x==0) return 0;
    else
        if (x % 2==0)
            return x%10 + 10*T(x/10);
        else return T(x/10);
}

void M(int i)
{
    long x;
    if (i<=n){
        cin>>x; M(i+1);
        cout<<T(x)<<' ';
    }
}

```

19. Se consideră o matrice pătratică a cu n linii și n coloane, ale cărei elemente sunt numere naturale. Se dorește permutarea circulară a elementelor de pe fiecare linie cu k poziții spre stânga. Realizați un subprogram recursiv M care efectuează această operație. Acesta primește, printr-un parametru, indicele liniei care urmează a fi prelucrată. În cadrul acestuia se va face apel la subprogramul recursiv Inv , care inversează elementele situate pe linia l între coloanele i și j . Toate cele trei valori sunt primite prin intermediul parametrilor.

Tabloul a și variabila n se consideră declarate global.

Exemplu:

Pentru $n=4$, $x=3$ și tabloul a	se va afișa
1 2 3 4	4 1 2 3
2 3 4 5	5 2 3 4
3 4 5 6	6 3 4 5
4 5 6 7	7 4 5 6

Soluție:

Considerăm tabloul unidimensional cu $n=5$ elemente (1, 2, 3, 4, 5). Permutarea circulară cu k poziții spre stânga o putem realiza prin trei operații de inversare a unor subtablouri.

- inversarea primelor k elemente;
- inversarea tuturor celor n elemente;
- inversarea primelor $n-k$ elemente.

În aceste condiții, considerând $k=2$, tabloul va suferi următoarele prelucrări:

2 1 3 4 5, 5 4 3 1 2, 3 4 5 1 2.

După cum se observă, vectorul a fost permutat cu k poziții spre stînga.

Aceasta este metoda prin care subprogramul M va permuta elementele fiecărei linii. Cele două valori primite de subprogram prin parametri i și k vor reprezenta în ordine: indicele liniei ce va fi prelucrată, respectiv numărul de poziții cu care se vor permuta elementele. Subprogramul M va fi implementat în Pascal ca procedură, iar în C++, ca funcție de tip `void`.

```

1 procedure Inv(l,i,j:byte);
2 var x:integer;
3 begin
4   if i<=j then begin
5     x:=a[l,i];a[l,i]:=a[l,j];
6     a[l,j]:=x;
7     Inv(l,i+1,j-1)
8   end;
9 end;
10
11 procedure M(i,k:byte);
12 begin
13   if i<=n then begin
14     Inv(i,1,k); Inv(i,1,n);
15     Inv(i,1,n-k); M(i+1,k);
16   end;
17 end;
```

```

void Inv(int l,int i,int j)
{int x;
 if (i<=j){
  x=a[l][i];a[l][i]=a[l][j];
  a[l][j]=x;
  Inv(l,i+1,j-1);
 }
}

void M(int i,int k)
{
 if (i<=n){
  Inv(i,0,k-1);
  Inv(i,0,n-1);
  Inv(i,0,n-k-1);
  M(i+1,k);
 }
}
```

156

20. Considerăm un număr natural x de cel mult 9 cifre. Realizați un program care îl scrie pe x ca sumă de termeni distincți din șirul lui *Fibonacci*. Primii termeni ai șirului sunt 1, 1, 2, 3, 5, 8, 13,..... În cadrul programului, vor fi definite două subprograme recursive:

- funcția *Fib*, care returnează termenul din șirul *Fibonacci* cu numărul de ordine transmis printr-un parametru.
- subprogramul *Gen*, care primește două valori naturale prin parametri v și n . Acesta afișează, în cadrul fiecărui autoapel, cel mai mare termen al șirului *Fibonacci* mai mic sau egal cu valoarea v . Numărul de ordine al acestui termen este dat de valoarea lui n . Datorită autoapelurilor, subprogramul *Gen* va afișa în final descompunerea numărului x în termeni distincți ai șirului lui *Fibonacci*.

Exemplu: Pentru numărul $x=80$, se va afișa 55, 21, 3, 1

Soluție:

Termenii șirului respectă regula $f_1=1$, $f_2=1$, $f_n=f_{n-1}+f_{n-2}$. Această recurență este implementată în manieră recursivă în cadrul funcției *Fib*.

Subprogramul *Gen* determină, în cadrul fiecărui autoapel, cel mai mare termen mai mic sau egal cu valoarea parametrului v . Subprogramul apelează funcția *Fib*. Condiția de oprire din recursivitate a subprogramului este îndeplinită când valoarea lui v devine 0. Un astfel de procedeu conduce la scrierea lui x ca sumă cu număr minim de termeni ai șirului.

```

1 var x:longint;
2
3 function Fib(n:byte):longint;
4 begin
5   if (n<=2) then fib:=1
6   else fib:=Fib(n-1)+Fib(n-2);
7 end;
8
9 procedure Gen(v,n:longint);
10 var y:longint;
11 begin
12   if Fib(n+1)<=v then
13     Gen(v,n+1)
14   else begin
15     y:=Fib(n);
16     write(y,' ');
17     if v-y>0 then Gen(v-y,0);
18   end;
19 end;
20
begin
  readln(x);
  Gen(x,0)
end;
```

```

#include<iostream.h>
long x;

long Fib(int n){
  if (n<=2) return 1;
  else return
    Fib(n-1)+Fib(n-2);
}

void Gen(long v,long n)
{long y;
 if (Fib(n+1)<=v)
  Gen(v,n+1);
 else {
  y=Fib(n);
  cout<<y<<' ';
  if (v-y>0) Gen(v-y,0);
 }
}

main(){
  cin>>x;
  Gen(x,0);
}
```

157

2.2.3 Probleme propuse

1. Se consideră un șir de n valori naturale de cel mult 9 cifre. Determinați numărul de valori care pot fi scrise sub forma $k!$ ($1*2*3*...*k$). Nu se vor folosi date structurate pentru reținerea celor n numere. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Ok*, care verifică dacă o valoare primită printr-un parametru reprezintă factorialul unei valori. Funcția returnează, în Pascal, valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- funcția *Nr*, care permite citirea celor n numere și returnează numărul de valori factoriale. În cadrul ei se va apela funcția *Ok*.

Exemplu: Pentru $n=5$ și valorile 7, 16, 6, 13, 24, se va afișa 2, deoarece $6=1*2*3$, $24=1*2*3*4$.

2. Considerăm un tablou unidimensional a cu $n(n<50)$ elemente numere naturale. Realizați un program care afișează, pe linii, elementele din vector grupate după cifra dominantă (prima în scrierea zecimală). Pe aceeași linie vor fi scrise elemente cu aceeași cifră dominantă. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Cif*, care determină cifra dominantă a unei valori primite printr-un parametru.
- subprogramul *Afis*, care afișează pe o linie a ieșirii standard elementele din vector ce au cifra dominantă egală cu o valoare transmisă prin parametrul c . Parametrul întreg i va indica poziția(indicele) elementului curent din vector.

Exemplu: Pentru $n=7$ și $a=(334, 124, 21, 34, 122, 1, 39)$, se va afișa:

124 122 1
21
334 34 39

3. Considerăm un tablou unidimensional a cu $n(n<50)$ elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care ordonează elementele vectorului, crescător după numărul de cifre pare pe care le conțin. În situația elementelor cu același număr de cifre pare, ordonarea se va face în funcție de valorile acestora. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Nr*, care determină numărul de cifre pare al unei valori primite printr-un parametru.
- subprogramul *Sortare*, care efectuează ordonarea crescătoare a elementelor vectorului, după regula dată. Acesta va face apel la funcția *Nr* și nu va conține instrucțiuni repetitive.

Exemplu: Pentru $n=7$ și $a=(384, 824, 21, 34, 122, 1268, 39)$, se va afișa:
39, 21, 34, 122, 384, 824, 1268,

4. Considerăm un tablou unidimensional a cu $n(n<50)$ elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care permite ștergerea elementului care are cel mai mare produs al cifrelor impare ce apar în scrierea sa. Dacă există mai multe elemente cu această proprietate, se va elimina cel de indice minim.

În cadrul programului, se vor defini două subprograme recursive:

- funcția *P*, care determină produsul cifrelor impare ale unei valori primite printr-un parametru.
- subprogramul *Del*, care permite ștergerea unui element dintr-un vector, al cărui indice este transmis prin parametrul întreg i . Subprogramul va avea doi parametri referință, reprezentând tabloul din care se efectuează ștergerea și lungimea acestuia. Ștergerea se va face prin deplasarea elementului $a[i+1]$ pe poziția i , ș.a.m.d.

Exemplu: Pentru $n=7$ și $a=(384, 824, 21, 349, 122, 1268, 37)$, se va afișa vectorul cu șase elemente: (384, 824, 21, 122, 1268, 37).

5. Considerăm un tablou unidimensional a cu $n(n<50)$ elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care determină cel mai mare divizor comun al elementelor care nu sunt numere prime. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Ok*, care verifică dacă o valoare primită printr-un parametru reprezintă un număr prim. Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- funcția *Cmmmd*, care returnează cel mai mare divizor comun a două valori transmise prin intermediul unor parametri valoare.

Exemplu: Pentru $n=7$ și $a=(37, 40, 13, 60, 31, 11, 140)$, se va afișa 20.

6. Considerăm un tablou unidimensional a cu $n(n<50)$ elemente numere naturale de cel mult 9 cifre. Să se realizeze un program care înlocuiește fiecare element cu valoarea lui din care s-au eliminat toate aparițiile cifrei maxime. Astfel, elementul 34241 va deveni 321. În cadrul programului, se vor defini trei subprograme recursive:

- funcția *Cmax*, care returnează cifra maximă a unei valori primite printr-un parametru întreg-lung.
- funcția *DelCif*, care primește prin parametrii x și c , două valori întregi. Aceasta returnează valoarea obținută din x prin ștergerea cifrei c .
- subprogramul *S*, care parcurge elementele vectorului primit printr-un parametru referință și le înlocuiește cu valorile ce respectă cerința din enunț. Subprogramul primește indicele elementului curent printr-un parametru valoare.

Exemplu: Pentru $n=7$ și $a=(37, 443, 13, 160, 31, 11, 140)$, se va afișa
 $a=(3, 3, 1, 10, 1, 0, 10)$.

7. Considerăm un tablou unidimensional a cu $n(n < 100)$ elemente numere naturale de cel mult 4 cifre. Să se realizeze un program care înlocuiește fiecare element cu cel mai apropiat palindrom de acesta. Astfel elementul 341 va deveni 343, iar 146 va deveni 141. În cadrul programului, se vor defini două subprograme recursive:

- funcția *Pal*, care verifică dacă o valoare primită prin parametru x reprezintă un număr palindrom. Parametrul referință z va memora, în final, inversul numărului x . Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- funcția *Search*, care primește prin parametrii x și t , două valori întregi. Aceasta returnează cel mare palindrom mai mic ca x sau cel mai mic palindrom mai mare ca x , după cum valoarea la apel a parametrului t este -1 sau 1. Subprogramul va face apel la funcția *Pal*.

Exemplu: Pentru $n=4$ și $a=(37, 44, 130, 16)$, se va afișa $a=(33, 44, 131, 11)$.

8. Considerăm un tablou unidimensional a cu $n(n < 100)$ elemente numere naturale de cel puțin două cifre. Să se realizeze un program care înlocuiește elementul cu număr maxim de apariții din vector cu suma primelor două cifre ale sale. În cadrul programului, se vor defini trei subprograme recursive:

- funcția *Sc*, care returnează suma primelor două cifre ale unei valori primite printr-un parametru întreg.
- funcția *Ap*, care primește, prin parametrii x și i , două valori întregi. Aceasta determină numărul de apariții al lui x în vectorul a . Valoarea transmisă prin parametrul i indică poziția curentă în vector.
- subprogramul *Max*, care primește, prin parametrul i , indicele elementului curent și returnează, prin parametrul referință x , elementul cu număr maxim de apariții în vectorul a . Subprogramul apelează funcția *Ap*.

Exemplu: Pentru $n=7$ și $a=(3700, \underline{544}, 130, \underline{544}, 3700, \underline{544}, 130)$, se va afișa $a=(3700, 9, 130, 9, 3700, 9, 130)$.

9. Considerăm un tablou unidimensional a cu $n(n < 100)$ elemente numere naturale de cel mult patru cifre. Să se realizeze un program care șterge toate aparițiile elementului minim. În cadrul programului, se vor defini trei subprograme recursive:

- funcția *Min*, care determină elementul minim al vectorului a . Parametrul i indică poziția curentă în vector.
- subprogramul *Del*, care permite ștergerea elementului din a al cărui indice este transmis prin parametrul i . Subprogramul va avea și doi parametri referință reprezentând tabloul din care se efectuează ștergerea și lungimea acestuia. Ștergerea se va face prin deplasarea elementului $a[i+1]$ pe poziția i , ș.a.m.d.

Exemplu: Pentru $n=7$ și $a=(37, \underline{4}, 130, \underline{4}, 37, \underline{4}, 130)$, se va afișa 37, 130, 37, 130.

10. Se consideră un vector ce conține n cifre zecimale ($n < 10$). Să se verifice dacă numerele formate cu cifrele din vector citite de la dreapta la stânga și de la stânga la dreapta reprezintă cuburi perfecte.

În cadrul programului, se vor defini două subprograme recursive:

- funcția *Nr*, care determină numărul format cu elementele vectorului a . Parametrul i indică poziția curentă în vector, iar parametrul t sensul de parcurgere al tabloului. Dacă la apel $t=-1$, atunci numărul va fi format cu cifrele vectorului de stînga la dreapta și de la dreapta spre stînga, dacă $t=1$.
- funcția *Cub*, care primește două valori întregi prin intermediul parametrilor x și i . Aceasta verifică există o valoare i astfel încât x se poate scrie ca i^3 . Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.

Exemplu: $n=3$ și vectorul: (1, 2, 5), se va afișa „125 este cub perfect, 521 nu este un cub perfect”.

11. Se consideră o valoare n naturală, de cel mult 9 cifre. Să se descompună n în toate modurile posibile ca sumă de două numere oglindite.

În cadrul programului, se vor defini două subprograme recursive:

- funcția *Inv*, care determină inversul unui număr transmis parametrul valoare x . Parametrul referință y va memora inversul construit succesiv.
- subprogramul *S*, care primește două valori întregi prin intermediul parametrilor x și i . Acesta afișează toate valorile i cu proprietatea că suma dintre i și inversul său este egală cu x .

Exemplu: Pentru $n=787$, se va afișa: 146+641, 245+542, 344+443.

12. Considerăm un șir de n valori naturale ($n \leq 50$) reținute în tabloul unidimensional a . Realizați un program care construiește alți doi vectori ce vor conține elementele lui a care sunt numere perfecte, respectiv pe cele care nu sunt numere perfecte. Un număr egal cu suma divizorilor săi strict mai mici decât el se numește număr perfect.

În cadrul programului, se vor defini două subprograme recursive:

- funcția *Sd*, care returnează numărul de divizori ai unei valori întregi transmise ca parametru.
- subprogram recursiv *Make*, care construiește alți doi vectori ce vor conține elementele lui a care sunt perfecte, respectiv cele care nu reprezintă valori perfecte. Subprogramul va returna vectorii construiți și lungimile acestora prin intermediul parametrilor. Acesta va apela funcția recursivă *Sd*.

Exemplu: Pentru $n=5$ și $a=(10, 6, 21, 28, 496)$, programul va construi doi vectori de lungime 3 respectiv 2: (6, 28, 496) și (10, 21).

13. Se citește, cu ajutorul unui șir de caractere, o expresie ai căror operanzi se consideră a fi doar parantezele rotunde. Astfel, o valoare inclusă între paranteze rotunde va fi convertită în baza 2. Să se afișeze valorile obținute după conversia în baza 2 a tuturor operanzilor prezenți în expresie.

În cadrul programului, se vor defini două subprograme recursive:

- subprogramul *B2*, care afișează cifră cu cifră valoarea obținută în urma conversiei în baza 2 a unui număr întreg transmis ca parametru.

- subprogramul *Extrag*, care primește, prin intermediul parametrului *s*, un șir de caractere reprezentând o expresie descrisă anterior. Subprogramul afișează valorile operanzilor obținute în urma conversiei în baza 2. Se va face apel la funcția *B2*.

Exemplu : Pentru expresia $(7)(8)(13)$, se va afișa 111, 1000, 1101.

14. Se citesc de la tastatură n cuvinte formate din cel mult 50 de litere ale alfabetului englez și o secvență s de caractere considerată o silabă. Realizați un program care afișează cuvintele care conțin cele mai multe silabe egale cu s . Nu se vor folosi date structurate pentru reținerea celor n cuvinte. În cadrul programului, vor fi definite următoarele două subprograme recursive:

- funcția *Nr*, care primește două șiruri de caractere prin parametrii x și y . Aceasta determină numărul de apariții a lui x în y .
- subprogramul *Solve*, care permite citirea celor n cuvinte și afișarea celor cu proprietatea specificată în enunț. Subprogramul va avea trei parametri: parametrul valoare i , care indică numărul de ordine al cuvântului care va fi citit, parametrul valoare s , care reprezintă silaba căutată și parametrul referință max , ce va memora numărul maxim de apariții al lui s printre cuvintele citite.

Exemplu : Pentru $n=4$, silaba *ma* și cuvintele *mama*, *tamara*, *mamaie*, *inca*, se va afișa *mamaie*, *mama*

15. Se citesc de la tastatură n cuvinte formate din cel mult 50 de litere ale alfabetului englez. Să se realizeze un program care afișează, în ordinea inversă citirii, cuvintele oglindite, din care lipsesc vocalele. Nu se vor folosi date structurate pentru reținerea celor n cuvinte. În cadrul programului, vor fi definite următoarele două subprograme recursive:

- funcția *Inv*, care primește un șir de caractere prin parametrul x și returnează inversul șirului din care lipsesc vocalele.
- subprogramul *Solve*, care permite citirea celor n cuvinte și afișarea în ordine inversă a valorilor cerute prin enunț. Subprogramul va avea un parametru valoare i , care indică numărul de ordine al cuvântului care va fi citit. Acesta va apela funcția *Inv*.

Exemplu : Pentru $n=4$ și cuvintele *mama*, *tamara*, *mamaie*, *inca*, se va afișa *cn*, *mm*, *rnt*, *mm*.

16. Se citesc de la tastatură n numere de cel mult patru cifre. Să se realizeze un program care determină produsul numerelor care sunt egale cu suma factorialelor cifrelor componente. Nu se vor folosi date structurate pentru reținerea celor n numere. În cadrul programului, vor fi definite următoarele trei subprograme recursive:

- funcția *Fact*, care primește o cifră prin parametrul x și returnează valoarea $x!$
- funcția *Sum*, care primește un număr întreg prin parametrul x și returnează suma factorialelor cifrelor lui x . Aceasta va face apel la funcția *Fact*.

- funcția *Prod*, care permite citirea celor n valori întregi și care returnează produsul valorilor cerute.

Exemplu: Pentru $n=5$ și numerele 1, 63, 2, 145, 496, se va afișa 290, deoarece $290 = 1 \cdot 2 \cdot 145$.

17. Considerăm șirul 0, 2, 1, 3, 2, 4, 3, 5, 4, 6.... Să se realizeze cu program care permite memorarea primilor n^2 termeni ai șirului, într-o matrice pătratică cu n linii și n coloane ($n < 10$). Termenii vor fi plasați în ordine pe linii de la stânga la dreapta. În cadrul programului, se vor defini următoarele două subprograme recursive:

- funcția recursivă *T*, care returnează termenul șirului cu numărul de ordine transmis prin intermediul unui parametru întreg.
- subprogramul recursiv *M*, care permite memorarea primilor n^2 termeni ai șirului, într-o matrice pătratică cu n linii și n coloane ($n < 10$). Subprogramul *M* primește două valori naturale prin parametrii i și j , valori care reprezintă indicii liniei și coloanei elementului curent. Matricea creată este returnată prin parametrul referință a .

Exemplu: Pentru $n=4$, programul va afișa matricea:

```
0 2 1 3
2 4 3 5
4 6 5 7
6 8 7 9
```

18. Fie un tablou bidimensional $A(n,m)$. Realizați un program care inversează elementele de pe liniile care încep cu un număr prim. În cadrul programului, vor fi definite următoarele subprograme recursive:

- funcția *Ok*, care verifică dacă o valoare primită printr-un parametru reprezintă un număr prim. Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- subprogramul *Inv*, care primește doi parametri întregi i și l . Acesta inversează elementele de pe linia l prin interschimbarea elementul $a[l,i]$ cu cel simetric în cadrul liniei.
- subprogramul *Pl*, care parcurge fiecare linie a tabloului și inversează liniile care încep cu un număr prim. El va primi, printr-un parametru valoare, indicele liniei curente.

Exemplu: Pentru $n=3$, $m=4$ și matricea:

4 2 3 7	se va afișa :	4 2 3 7
2 3 4 5		5 4 3 2
3 4 5 2		2 5 4 3

19. Se consideră un tablou bidimensional cu n linii și n coloane ($1 \leq n \leq 100$), având componente de tip întreg. Cele două diagonale ale tabloului împart tabloul în patru regiuni în formă de triunghi. Se cere să se determine suma componentelor din interiorul fiecărei zone. În cadrul programului, vor fi definite patru funcții

recursive. Fiecare dintre ele calculează suma elementelor situate într-una dintre cele patru zone. Toate funcțiile vor primi, prin doi parametri valoare, indicii liniei și elementului curent. În cadrul nici unei funcții nu se vor folosi instrucțiuni repetitive.

Exemplu: Pentru $n=5$ și tabloul:

0 1 1 1 0	se va afișa:
2 0 1 0 3	S1=4
2 2 0 3 3	S2=8
2 0 4 0 3	S3=12
0 4 4 4 0	S4=16

20. Să se rearanjeze elementele unei matrice de dimensiune $n \times m$, astfel încât ele să fie ordonate crescător atât pe linii cât și pe coloane. În cadrul programului, vor fi definite următoarele subprograme recursive:

- Subprogramul *OrdL*, care ordonează elementele situate pe o linie al cărui indice este transmis printr-un parametru valoare. Subprogramul nu conține nici o structură repetitivă.
- Subprogramul *OrdC*, care ordonează elementele situate pe o coloană al cărui indice este transmis printr-un parametru valoare. Subprogramul nu conține nici o structură repetitivă.

Exemplu: $n=3$, $m=4$ și matricea

3 1 8 9	se va afișa:
4 6 5 7	0 1 2 3
2 0 1 3	1 3 6 7
	4 5 8 9

21. Se consideră tabloul bidimensional a cu n linii și n coloane ($1 \leq n \leq 100$), având componente de tip întreg. Realizați un program care înlocuiește fiecare element situat sub diagonala secundară cu suma exponenților ce apar la descompunerea lui în factori primi. De exemplu valoarea $360=2^3 \cdot 3^2 \cdot 5^1$ va fi înlocuită cu numărul $6(3+2+1)$. În cadrul programului, se vor defini următoarele subprograme recursive:

- funcția *S_e*, care returnează suma exponenților din descompunerea în factori primi a unui număr primit prin parametrul întreg x .
- subprogramul *Diag*, care parcurge toate elementele situate sub diagonala principală a lui a . Acestea vor fi înlocuite cu valori cerute în enunț. Subprogramul va avea doi parametri valoare, reprezentând indicele liniei și al coloanei elementului curent. Nu se vor folosi instrucțiuni repetitive.

Exemplu: $n=4$ și matricea

123 211 213 901	se va afișa:
124 216 215 360	123 211 213 901
122 125 125 18	124 216 215 6
535 400 625 120	122 125 3 3
	535 6 4 5

22. Se citește de la tastatură o valoare naturală de cel mult 5 cifre. Să se realizeze un program care permite memorarea primelor n^2 numere prime mai mici decât k , într-o matrice pătratică cu n linii și n coloane. Programul va determina cea mai mare valoare posibilă pentru n . Termenii vor fi plasați în ordine pe linii de la stânga la dreapta. În cadrul programului, se vor defini următoarele două subprograme recursive:

- funcția *T*, care verifică dacă o valoare naturală transmisă printr-un parametru reprezintă un număr prim. Funcția returnează în Pascal valoarea *True* sau *False*, respectiv o valoare nenulă sau 0 în C++.
- subprogram *M*, care permite memorarea primelor n^2 numere prime într-o matrice pătratică cu n linii și n coloane. Subprogramul primește trei valori naturale prin parametrii i , j și v , valori care reprezintă indicii liniei și coloanei elementului care va memora numărul v , doar dacă acesta este prim. Se va face apel la funcția *Ok*.

Exemplu: Pentru $k=30$, programul va determina $n=3$ și va genera matricea:

2 3 5
7 11 13
17 19 23

23. Realizați un program care generează toate cuvintele de lungime n formate din cele două caractere ale codului Morse '.', '-'. În cadrul programului, se va defini subprogramul recursiv *Morse* care generează și afișează, pe câte o linie a ieșirii standard, șirurile de caractere cerute. Subprogramul va avea un singur parametru valoare de tip șir de caractere.

Exemplu: Pentru $n=2$ se va afișa: .. .- -. --

24. Se consideră un tablou unidimensional de lungime n . Asupra acestuia se vor efectua operații de "tăiere" care constau în "înjumătățirea" lui și îndepărtarea unuia dintre cele două subtablouri astfel obținute. Dacă n este impar, elementul de pe poziția $[n/2]$ este eliminat. Procesul de tăiere se repetă până la obținerea unui tablou cu un singur element. Să se afișeze, pe o singură linie în fișierul text *Out.txt*, toate elementele care se pot obține la încheierea procesului de tăiere.

Datele de intrare se citesc din fișierul *In.txt* în următorul format: Pe prima linie numărul n , iar pe următoarea linie elementele separate prin câte un spațiu.

Exemplu: Pentru fișierul *In.txt*

7	Out.txt
1 2 3 4 5 6 7	1 3 5 7

2.3. Probleme de concurs

2.3.1 Probleme rezolvate

1. (**Partiție- *****) Se consideră un vector cu $N \leq 10.000$ componente numere întregi, cu valori cuprinse între 1 și 50000. Să se partiționeze acest vector în cât mai puține subșiruri strict crescătoare.

Pe prima linie a fișierului de intrare *partitie.in* se găsește valoarea N . Următoarea linie conține N numere separate prin câte un singur spațiu.

Prima linie a fișierului *partitie.out* va conține numărul de subșiruri strict crescătoare.

Exemplu:

<i>partitie.in</i>	3	<i>partitie.out</i>
2 1 3 4 0 7 2 7 10		

Soluție:

Se baleiază vectorul de la stânga la dreapta, pentru a se construi subșirurile cu o singură parcurgere. Fiecare element este adăugat la sfârșitul unuia dintre subșirurile deja formate sau, dacă acest lucru nu este posibil, el este pus separat pentru a începe un nou subșir. În situațiile în care un element poate fi alipit la mai multe subșiruri, se preferă alipirea la subșirul care se termină într-un număr *cât mai mare*. În concluzie, pentru fiecare element $V[i]$ trebuie să se afle, dintre toate subșirurile deja create, capătul cu valoarea cea mai mare, dar strict mai mică decât $V[i]$. Pentru aceasta se recomandă menținerea valorilor de la capetele subșirurilor într-un vector separat, sortat, asupra căruia se pot efectua căutări binare. De asemenea, se va reține pentru fiecare element din V succesorul său în subșirul crescător din care face parte.

```

1 var n,nv,i,x,p:integer; v:array[0..10000] of integer;
2
3 function caut_binar(var x:integer):integer;
4 var st,dr,mij:integer;
5 begin
6   st:=1; dr:=nv+1;
7   while st<dr do begin
8     mij:=(st+dr) div 2;
9     if v[mij]>=x then st:=mij+1 else dr:=mij;
10  end; caut_binar:=st;
11 end;
12
13 begin
14   assign(input,'partitie.in');reset(input); read(n);
15   for i:=1 to n do begin
16     read(x); p:=caut_binar(x);
17     v[p]:=x;
18     if p=nv+1 then inc(nv);
19   end;
20   assign(output,'partitie.out'); rewrite(output);
21   writeln(nv);
22 end.
```

166

```

1 #include <stdio.h>
2 int n,v[10000],nv;
3
4 int caut_binar(int x) {
5   int st,dr,mij;
6   for (st=0,dr=nv;st<dr;){
7     mij=(st+dr)/2;
8     if (v[mij]>=x) st=mij+1; else dr=mij;
9   }
10  return st;
11 }
12
13 void main() {
14   int i,x,p;
15   freopen("partitie.in","r",stdin); scanf("%d",&n);
16   for (i=0;i<n;i++){
17     scanf("%d",&x);
18     p=caut_binar(x);
19     v[p]=x;
20     if (p==nv) nv++; }
21   freopen("partitie.out","w",stdout);
22   printf("%d\n",nv);
23 }
```

2. (**Distanța - *****) B.A. vă da $N \leq 10.000$ puncte în spațiu, cu coordonate numere întregi. Vă cere distanța Manhattan dintre cele mai depărtate 2 puncte (tot folosind distanța Manhattan). Reamintim că distanța Manhattan dintre 2 puncte în spațiu cu coordonatele $(X1, Y1, Z1)$ și respectiv $(X2, Y2, Z2)$ este: $|X1 - X2| + |Y1 - Y2| + |Z1 - Z2|$.

Pe prima linie a fișierului *distanța.in* se află N , reprezentând numărul de puncte. Pe următoarele N linii sunt câte 3 numere întregi separate printr-un spațiu, reprezentând în ordine, coordonatele X, Y și respectiv Z ale celor N puncte ($-1.000.000 \leq X, Y, Z \leq 1.000.000$).

Fișierul *distanța.out* va conține o linie cu un singur număr întreg, și anume distanța cea mai mare între 2 puncte dintre cele N .

Exemplu:

<i>distanța.in</i>	3	<i>distanța.out</i>
8		
0 0 0		
0 0 1		
0 1 0		
1 0 0		
1 1 0		
1 0 1		
0 1 1		
1 1 1		

Soluție:

Pentru a rezolva problema trebuie explicitate modulele. Se știe că $|X| = X$, dacă $X \geq 0$ sau $-X$ dacă $X < 0$. În același fel se poate scrie și $|X1 - X2| + |Y1 - Y2| + |Z1 - Z2|$. Cele 3 module se pot explicita în 8 moduri. Pentru fiecare mod se obține un alt

167

punct, corespunzător acestei dezvoltări a modulelor. Pentru fiecare dintre aceste 8 direcții (corespunzătoare celor 8 explicitări ale modulelor) nu trebuie decât să se rețină minimul și maximum. La sfârșit se verifică pe care dezvoltare s-a obținut cea mai mare distanță. Datorită dimensiunii datelor, se vor folosi compilatoarele FreePascal sau GCC.

```

1 var n, rez, i, j, t, sx, sy, sz, min, max: longint;
2   x, y, z: array[1..10000] of longint;
3
4 begin
5   assign(input, 'distanța.in'); reset(input);
6   readln(n);
7   for i:=1 to n do read(x[i], y[i], z[i]);
8   for sx:=-1 to 1 do
9     for sy:=-1 to 1 do
10      for sz:=-1 to 1 do
11        if (sx<>0) and (sy<>0) and (sz<>0) then begin
12          min:=2000000000; max:=-2000000000;
13          for j:=1 to n do begin
14            t:=sx*x[j]+sy*y[j]+sz*z[j];
15            if min>t then min:=t;
16            if max<t then max:=t;
17          end;
18          if rez<max-min then rez:=max-min;
19        end;
20   assign(output, 'distanța.out'); rewrite(output);
21   writeln(rez);
22 end.
```

```

1 #include <stdio.h>
2 long n, X[10000], Y[10000], Z[10000], rez=-2000000000;
3
4 void main() {
5   long i, j, t, sx, sy, sz, min, max;
6   freopen("distanța.in", "r", stdin);
7   scanf("%ld", &n);
8   for (i=0; i<n; i++)
9     scanf("%ld %ld %ld", X+i, Y+i, Z+i);
10  for (sx=-1; sx<=1; sx++)
11    for (sy=-1; sy<=1; sy++)
12      for (sz=-1; sz<=1; sz++)
13        if (sx&&sy&&sz) {
14          min=2000000000, max=-2000000000;
15          for (j=0; j<n; j++) {
16            t=sx*X[j]+sy*Y[j]+sz*Z[j];
17            if (min>t) min=t;
18            if (max<t) max=t;
19          }
20          if (rez<max-min) rez=max-min;
21        }
22  freopen("distanța.out", "w", stdout);
23  printf("%lld\n", rez);
24 }
```

168

3. (Maraton - **) $N \leq 3.000$ sportivi numerotați de la 1 la N iau parte la un maraton. Clasamentul final este codificat sub forma unui vector A de lungime N . Fiecare element $A[i]$ din vector are următoarea interpretare: concurentul clasat pe locul i a devansat un număr de $A[i]$ concurenți ale căror numere de pe tricou sunt mai mari decât al lui. În decursul ultimului an, toți acești N sportivi au participat la $M \leq 10$ probe de maraton și de fiecare dată au avut același număr pe tricoul de concurs. Toate clasamentele finale au fost codificate după regula descrisă.

Știind că toți sportivii au terminat fiecare dintre cele M probe de maraton, aflați care concurenți au evoluat din ce în ce mai bine, adică, la fiecare nouă probă, locul pe care l-au ocupat a fost strict mai mic decât la proba anterioară.

În fișierul text *maraton.in*, pe prima linie, se află două numere naturale N și M , despărțite printr-un spațiu. Pe următoarele M linii sunt scrise, în ordine cronologică a momentului desfășurării, clasamentele finale (câte un clasament pe o linie). Numerele scrise pe aceeași linie sunt despărțite prin câte un spațiu.

În fișierul text *maraton.out*, pe o singură linie, se vor scrie în ordine crescătoare, numerele de pe tricou ale concurenților identificați cu evoluții ascendente. În cadrul liniilor, numerele vor fi despărțite prin câte un spațiu. Dacă nu există soluție, fișierul de ieșire va conține mesajul NU EXISTA.

Exemplu:

<i>maraton.in</i>	<i>maraton.out</i>
5 2	1 4
3 2 2 1 0	
3 3 1 1 0	

Soluție:

Algoritmul construiește fiecare clasament astfel: se pornește cu elementul $A[1]$ care indică tricoul primului clasat: $N-A[1]$, selectându-se acest număr. Se continuă identificarea ordinii la sosire ignorându-se numerele de tricou deja selectate. La finalul fiecărei probe, se reține, într-un vector, poziția în clasament a fiecărui concurent, marcându-se cei care nu au avut o evoluție ascendentă.

```

1 type sir=array[1..3005] of word; sirb=array[1..3005] of boolean;
2 var i, n, nr, k, j, m: longint; sel: sirb; p, a: sir; ok: boolean;
3
4 procedure rezolva;
5 var i, j: integer;
6 begin
7   for i:=1 to n do begin
8     j:=n+1; nr:=0;
9     while (j>1) and (nr<=a[i]) do begin
10       dec(j); if not sel[j] then inc(nr);
11     end;
12     sel[j]:=true;
13     if p[j]=0 then p[j]:=i
14     else if (p[j]>i) and (p[j]<>n+1) then p[j]:=i
15     else p[j]:=n+1;
16   end;
17 end;
```

```

18 procedure citește_si_rezolvă;
19 begin
20   fillchar(a, sizeof(a), 0); fillchar(p, sizeof(p), 0);
21   assign(input, 'maraton.in'); reset(input); readln(n, m);
22   for k:=1 to m do begin
23     fillchar(sel, sizeof(sel), false);
24     for i:=1 to n do read(a[i]);
25     rezolvă;
26   end;
27 end;
28 procedure scrie;
29 begin
30   assign(output, 'maraton.out'); rewrite(output); ok:=true;
31   for i:=1 to n do
32     if p[i]<>n+1 then begin write(i); ok:=false; break; end;
33   for i:=i+1 to n do
34     if p[i]<>n+1 then write(' ', i);
35   if ok then write('NU EXISTA'); writeln;
36 end;
37
38 begin citește_si_rezolvă; scrie; end.

```

```

1 #include <stdio.h>
2 #include <string.h>
3 int n, m, p[3005], a[3005], ok; char sel[3005];
4 void rezolvă() {
5   int i, j, nr;
6   for (i=0; i<n; i++) {
7     for (j=n, nr=0; j>0&&nr<=a[i];)
8       if (!sel[--j]) nr++;
9     sel[j]=1; if (p[j]==-1) p[j]=i;
10    else if (p[j]>i&&p[j]!=n) p[j]=i; else p[j]=n;
11  }
12 }
13 void citește_si_rezolvă() {
14   int i, k;
15   memset(a, 0, sizeof(a)); memset(p, -1, sizeof(p));
16   freopen("maraton.in", "r", stdin); scanf("%d%d", &n, &m);
17   for (k=0; k<m; k++) {
18     memset(sel, 0, sizeof(sel));
19     for (i=0; i<n; i++) scanf("%d", &a[i]);
20     rezolvă();
21   }
22 }
23 void scrie() {
24   int i;
25   freopen("maraton.out", "w", stdout); ok=1;
26   for (i=0; i<n; i++)
27     if (p[i]!=n) { printf("%d", i+1); ok=0; break; }
28   for (i++; i<n; i++)
29     if (p[i]!=n) printf(" %d", i+1);
30   if (ok) printf("NU EXISTA");
31   putchar('\n');
32 }
33 void main() {
34   citește_si_rezolvă(); scrie();
35 }

```

4. (Prime - ***) Se dau $n \leq 9$ numere prime distincte p_1, p_2, \dots, p_n mai mici sau egale decât $k \leq 1.000.000.000$. Se cere să se determine câte numere naturale nenule mai mici sau egale cu k sunt divizibile cel puțin cu unul dintre numerele p_1, p_2, \dots, p_n . Prima linie a fișierului de intrare *prime.in* conține numărul n . Pe a doua linie sunt scrise numerele p_1, p_2, \dots, p_n , separate prin câte un spațiu, iar pe a treia linie se află numărul k . Se garantează că $p_1 \times p_2 \times \dots \times p_n \leq 2.000.000.000$.

În fișierul de ieșire *prime.out* se va scrie numărul numerelor naturale nenule mai mici sau egale cu k , divizibile cu cel puțin cu unul dintre numerele p_1, p_2, \dots, p_n .

Exemplu: *prime.in*

2
3 5
23

prime.out

10

(<http://campion.edu.ro>)

Soluție:

Se va folosi principiul includerii și excluderii, cu care se poate determina cardinalul reuniunii a mai multor mulțimi de numere.

$$\begin{aligned}
 \left| \bigcup_{i=1}^n A_i \right| &= \sum_{i=1}^n |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \\
 &+ \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n-1} \left| \bigcap_{i=1}^n A_i \right|
 \end{aligned}$$

Se vor considera mulțimile:

$$A_i = \{p_i * q \mid q \in \mathbb{N}^*, p_i * q \leq k\}$$

Numărul căutat va fi cardinalul reuniunii acestor mulțimi. Pentru intersecția mulțimilor se vor folosi formulele:

$$|A_i| = \left\lfloor \frac{k}{p_i} \right\rfloor, |A_i \cap A_j| = \left\lfloor \frac{k}{p_i * p_j} \right\rfloor, |A_i \cap A_j \cap A_l| = \left\lfloor \frac{k}{p_i * p_j * p_l} \right\rfloor, \dots$$

iar pentru generarea tuturor submulțimilor se vor utiliza numere binare ce vor reprezenta vectori caracteristici.

```

1 var n, i, j, nr: integer; k, rez, t: longint;
2   p: array[1..10] of integer;
3
4 begin
5   assign(input, 'prime.in'); reset(input);
6   read(n);
7   for i:=1 to n do read(p[i]); read(k);
8   for i:=1 to (1 shl n)-1 do begin
9     t:=k; nr:=0;
10    for j:=0 to n-1 do
11      if i and (1 shl j) > 0 then begin t:=t div p[j+1]; inc(nr); end;
12    if odd(nr) then inc(rez, t) else dec(rez, t);
13  end;
14  assign(output, 'prime.out'); rewrite(output);
15  writeln(rez);
16 end.

```

```

1 #include <stdio.h>
2 int n,p[10]; long k,rez;
3
4 void main() {
5     int i,j,nr; long t;
6     freopen("prime.in","r",stdin);
7     scanf("%d",&n);
8     for (i=0;i<n;i++) scanf("%d",p+i);
9     scanf("%ld",&k);
10    for (i=1;i<=(1<n);i++){
11        for (t=k,nr=j=0;j<n;j++)
12            if (i&(1<j)) t/=(long)p[j],nr++;
13        if (nr%2==1) rez += t;
14        else rez -= t;
15    }
16    freopen("prime.out","w",stdout);
17    printf("%ld\n",rez);
18 }

```

5. (Orar - **) Un laborator acceptă vizitatori de la firmele interesate de utilizarea aparaturii proprii doar două ore pe zi, într-un interval fix de zile. Fiecare firmă trebuie să-și determine propriul orar pentru utilizarea laboratorului exact două ore pe zi. Aceste două ore pot fi în prima perioadă a zilei, în a doua perioadă sau în a treia perioadă. Orarul este fixat, de la început, pentru toată perioada celor $N \leq 44$ zile și trebuie să respecte următoarele condiții:

- dacă o firmă selectează, într-o zi, a treia perioadă, atunci este obligată ca, în următoarea zi să selecteze prima perioadă;
- prima perioadă poate fi selectată numai dacă în ziua precedentă a fost selectată perioada a treia;

Pentru un număr de zile N dat, scrieți un program care calculează numărul posibilităților de întocmire a orarului. Fișierul de intrare *orar.in* conține o singură linie pe care se află numărul întreg N , care reprezintă numărul de zile. Fișierul de ieșire *orar.out* va conține o singură linie pe care se va afla numărul întreg care reprezintă numărul de posibilități de întocmire a orarului.

Exemplu:

orar.in	5	orar.out
4		

(BOI, 2001)

Soluție:

Se va construi o procedură recursivă $Nr(x)$, care determină numărul de posibilități pentru orar pentru x zile. Cum orice orar pe $x-1$ zile poate fi transformat într-un orar pe x zile alegând perioada a doua, iar orice orar pe $x-2$ zile poate fi transformat într-un orar pe x zile alegând perioada a treia și apoi prima, $Nr(x)$ va întoarce $Nr(x-1) + Nr(x-2)$. Procedura recursivă se oprește când se ajunge la unul din cazurile de bază $x=0$ sau $x=1$. Pentru a evita un număr mare de apeluri al proceduri cu aceiași parametrii de mai multe ori (fiind total ineficient), se va folosi o tehnică numită *memoizare*, care constă în stocarea rezultatului într-un vector de fiecare data când

se apelează procedura cu anumiți parametri, iar când procedura este apelată cu parametrii care au mai fost deja, se preia rezultatul din vector, evitând astfel calcule inutile. O renunțare totală la apeluri recursive și calcularea rezultatelor din vector într-o anumită ordine conduce la o rezolvare folosind metoda programării dinamice.

```

1 var n,i:integer; mem:array[0..44] of longint;
2
3 function Nr(x:integer):longint;
4 begin
5     if mem[x]<>-1 then begin Nr:=mem[x]; exit; end;
6     if (x=0) or (x=1) then mem[x]:=1
7     else mem[x]:=Nr(x-1)+Nr(x-2);
8     Nr:=mem[x];
9 end;
10
11 begin
12     assign(input,'orar.in'); reset(input); readln(n);
13     for i:=0 to n do mem[i]:=-1;
14     assign(output,'orar.out'); rewrite(output);
15     writeln(Nr(n));
16 end.

```

```

1 #include <stdio.h>
2 int n; long mem[45];
3 long Nr(int x) {
4     if (mem[x]==-1) return mem[x];
5     if (x==0 || x==1) mem[x]=1;
6     else mem[x]=Nr(x-1)+Nr(x-2);
7     return mem[x];
8 }
9
10 void main() {
11     int i;
12     freopen("orar.in","r",stdin); scanf("%d",&n);
13     for (i=0;i<=n;i++) mem[i]=-1;
14     freopen("orar.out","w",stdout);
15     printf("%lld\n", Nr(n));
16 }

```

6. (Oo - ***) Fermierul Ion are o fermă de formă circulară, unde cresc $N \leq 10.000$ găini. Ferma a fost împărțită în N sectoare, numerotate de la 1 la N , astfel încât oricare două sectoare având numere consecutive sunt adiacente (se află unul lângă altul). În plus, primul și ultimul sector sunt adiacente. În fiecare sector se află câte o găină, iar aceasta depune un anumit număr de ouă în fiecare zi. După ce găinile depun ouăle, fermierul Ion dorește să le adune, pentru a le mânca. Deoarece fermierul este foarte lacom, de fiecare dată el alege două sectoare adiacente din care adună ouăle simultan. Din păcate, din cauza lăcomiei sale, găinile din sectoarele vecine cu cele două alese se sperie și devin violente, motiv pentru care fermierul nu mai poate aduna ouăle din aceste sectoare. Determinați numărul maxim de ouă pe care le poate aduna fermierul Ion, în urma aplicării strategiei sale lacome.

Fișierul de intrare *oo.in* conține, pe prima linie, numărul de sectoare în care este împărțită ferma (și, implicit, numărul de găini). Pe următoarea linie se află N numere întregi din intervalul $[0,100]$, reprezentând numărul de ouă depuse de fiecare găina, în ordinea sectoarelor în care se află acestea. În fișierul *oo.out*, veți afișa numărul maxim de ouă pe care le poate aduna fermierul Ion.

Exemplu:

<i>oo.in</i>		<i>oo.out</i>
10		20
3 4 0 1 0 6 7 1 2 1		

("Stelele informaticii" 2003, București)

Soluție:

Se construiesc trei tablouri unidimensionale (conceptual; practic, se va folosi unul singur):

A_1 - elementul de pe poziția i reprezintă numărul maxim de ouă adunate până în sectorul i , dacă se adună ouăle din primele două sectoare, luate împreună

A_2 - elementul de pe poziția i reprezintă numărul maxim de ouă adunate până în sectorul i , dacă se adună ouăle din sectorul al doilea, împreună cu cele din al treilea

A_3 - elementul de pe poziția i reprezintă numărul maxim de ouă adunate până în sectorul i , dacă se adună ouăle din primul și ultimul sector, luate împreună.

Evident, în primul caz nu se pot lua ouăle din ultimul sector etc. Valorile din aceste tablouri se pot calcula pe baza unor recurențe simple, ceea ce conduce la un algoritm liniar. În final, se selectează maximum dintre aceste valori, determinate.

```

1 var n,i,j,k:integer; rez:longint;
2   oo:array[0..9999] of integer; a:array[0..9999] of longint;
3
4 function max(a,b:longint):longint;
5 begin
6   if a>b then max:=a
7   else max:=b
8 end;
9
10 begin
11   assign(input,'oo.in');reset(input); read(n);
12   for i:=0 to n-1 do read(oo[i]);
13   if n=2 then rez:=oo[0]+oo[1]
14   else
15     for k:=0 to 1 do begin
16       a[k] := 0; a[(k+1) mod n] := 0;
17       a[(k+2) mod n] := oo[(k+1) mod n] + oo[(k+2) mod n];
18       i:=(k+3) mod n;
19       while i<>k do begin
20         a[i]:=max(a[(i-1) mod n], a[(i-3) mod n]+oo[(i-1) mod n]+oo[i]);
21         i:=(i+1) mod n; end;
22       rez:=max(rez, a[(k-1+n) mod n]);
23     end;
24   assign(output,'oo.out');rewrite(output);
25   writeln(rez); end.
```

```

1 #include <stdio.h>
2 int n,oo[10000];long A[10000],rez;
3 long max(long a, long b) { return a > b ? a : b; }
4
5 void main() {
6   int i,j,k;
7   freopen("oo.in","r",stdin);
8   scanf("%d",&n);
9   for (i=0;i<n;i++) scanf("%d",&oo[i]);
10  if (n==2) rez = oo[0]+oo[1];
11  else
12    for (k=0; k<2; k=(k+1)%n) {
13      A[k] = 0;
14      A[(k+1)%n] = 0;
15      A[(k+2)%n] = oo[(k+1)%n] + oo[(k+2)%n];
16      for (i=(k+3)%n; i!=k; i=(i+1)%n)
17        A[i] = max(A[(i-1)%n], A[(i-3)%n]+oo[(i-1)%n]+oo[i]);
18      rez = max(rez, A[(k-1+n)%n]);
19    }
20  freopen("oo.out","w",stdout); printf("%ld\n", rez);
21 }
```

7. (**Ciobănașul** - ***) Ciobănașul Ion s-a decis să construiască un țarc nou pentru oile sale. Terenul pe care ciobănașul Ion vrea să construiască țarcul este de formă dreptunghiulară parcat în $n*m$ parcele și este puțin denivelat. Despre fiecare parcelă știm dacă se află la nivelul normal sau este denivelată. Țarcul trebuie să aibă o formă dreptunghiulară, laturile paralele cu laturile terenului și este susținut de patru țaruși situați în cele patru colțuri. Cei patru țaruși pot fi înfițiți în pământ numai în parcele nivelate. Pentru ciobănaș ar fi foarte important să știe câte amplasări posibile există pentru țarc, dar el e sătul de numărat oi, așa că vă cere ajutorul vostru pentru a număra dreptunghiuri.

Fișierul de intrare *ciobanas.in* conține pe prima linie două numere întregi $n \leq 250$ și $m \leq 2.000$ separate printr-un singur spațiu care reprezintă dimensiunile terenului. Fiecare dintre următoarele n linii conțin câte m numere, separate prin spațiu, care pot avea valorile 0 pentru teren denivelat sau 1 pentru nivel normal.

Fișierul de ieșire *ciobanas.out* trebuie să conțină o singură linie pe care se va afla un singur număr reprezentând numărul de posibilități de amplasare.

Exemplu:

<i>ciobanas.in</i>		<i>ciobanas.out</i>
3 3		5
0 1 1		
1 1 1		
1 1 1		

Soluție:

Pentru a determina numărul dreptunghiurilor ce au în vârfuri parcele nivelate se vor considera toate perechile de linii; pentru fiecare pereche se va determina numărul coloanelor ce conțin valoarea 1 în ambele linii – pentru c astfel de coloane se vor forma $c*(c-1)/2$ dreptunghiuri. Pentru a mări viteza de execuție se va păstra matricea în numere de câte 8 biți, astfel împărțindu-se fiecare rând în bucăți de 8; când se iau două rânduri va trebui aplicată operația AND pe biții rândurilor și apoi numărarea biților de 1.

```

1 var n,m,i,j,k:integer; nr:array[0..255] of integer;
2   a:array[0..250,0..250] of byte; rez,t:double;
3
4 procedure setbit(i,j:integer);
5 begin A[i,j div 8]:=A[i,j div 8] or 1 shl(j mod 8); end;
6
7 begin
8   for i:=0 to 255 do begin
9     j:=i;
10    while j>0 do begin
11      if j mod 2=1 then inc(nr[i]); j:=j div 2;
12    end;
13  end;
14  assign(input,'ciobanas.in'); reset(input); read(n,m);
15  for i:=0 to n-1 do begin
16    for j:=0 to m-1 do begin
17      read(k);
18      if k>0 then setbit(i,j);
19    end;
20    for j:=0 to i-1 do begin
21      t:=0;
22      for k:=(m div 8)+1 downto 0 do
23        t:=t+nr[a[i,k] and a[j,k]];
24      rez:=rez+t*(t-1)/2;
25    end;
26  end;
27  assign(output,'ciobanas.out'); rewrite(output);
28  writeln(rez:0:0);
29 end.

```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 int n,m,nr[256]; unsigned char far A[252][252]; double rez=0,t;
4
5 void set_bit(int i, int j) { A[i][j/8] |= 1<<(j%8); }
6
7 void main() {
8   int i,j,k;
9   for (i=0; i<256; i++)
10     for (j=i; j>0; j/=2) nr[i]+=j%2;
11   freopen("ciobanas.in", "r", stdin);
12   scanf("%d %d", &n, &m);
13   for(i=0; i<n; i++) {
14     for(j=0; j<m; j++) {
15       scanf("%d", &k);
16       if (k) set_bit(i, j);
17     }
18     for(j=0; j<i; j++) {
19       for(t=0, k=(m/8)+1; k>=0; k--)
20         t+=(double) nr[A[i][k]&A[j][k]];
21       rez+=t*(t-1)/2;
22     }
23   }
24   freopen("ciobanas.out", "w", stdout);
25   printf("%.01f\n", rez);
26 }

```

8. (Fotbal - ****) Ciobănașul Ion se plictisește păzind oile și s-a gândit să diversifice activitățile. El a început să învețe un grup de oi să joace fotbal și vrea să împartă aceste oi în două echipe și să organizeze un meci între ele, dar pentru ca meciul să fie cât mai interesant el a impus restricția ca echipele să aibă valori cât mai apropiate posibil (deoarece meciul de fotbal e între oi, nu trebuie ca numărul de oi dintr-o echipă să fie egal cu numărul de oi din cealaltă echipă, dar fiecare oaie care știe fotbal trebuie repartizată în una dintre echipe). Ciobănașul vrea să știe câte astfel de repartizări ale oilor în două echipe există astfel încât diferența de valoare dintre echipe să fie minimă și care este acea valoare.

Fișierul de intrare *fotbal.in* conține, pe prima linie, numărul $n \leq 24$ al oilor care știu să joace fotbal, iar pe a doua linie conține n numere întregi, separate între ele prin spații, care reprezintă valorile fotbalistice pentru fiecare oaie. Valoarea fotbalistică a unei oi este un număr întreg cuprins între 1 și 1000.

Fișierul de ieșire *fotbal.out* trebuie să conțină o singură linie pe care se vor afla două numere întregi separate între ele printr-un singur spațiu. Primul număr reprezintă diferența minimă care se poate obține între valorile celor două echipe, iar cel de-al doilea număr reprezintă numărul de repartizări în două echipe ale oilor astfel încât diferența dintre valorile celor două echipe să fie minimă.

Exemplu: *fotbal.in*

```

4
1 2 4 6

```

fotbal.out

```

1 2

```

Soluție:

Se vor lua în considerare toate submulțimile care se pot forma folosind primele $\lfloor n/2 \rfloor$ oi și se vor sorta în funcție de valoarea fotbalistică totală a oilor care fac parte dintr-o submulțime. În continuare, se vor determina toate submulțimile formate din ultimele $n - \lfloor n/2 \rfloor$ oi. Pentru fiecare astfel de submulțime A , se va căuta în lista primelor submulțimi (folosind căutarea binară) acea submulțime B care are valoarea cea mai apropiată de $S/2 - x$, unde S este valoarea fotbalistică însumată a tuturor oilor, iar x este valoarea fotbalistică totală a oilor din mulțimea A . Echipa "candidată" va fi obținută prin reuniunea mulțimilor A și B . Dacă valoarea fotbalistică a echipei este mai apropiată de $S/2$ decât cea mai bună soluție obținută anterior, atunci avem acum o nouă diferență minimă și o singură posibilitate (deocamdată) de a o obține. Dacă diferența este egală cu cea curentă, atunci se va crește numărul posibilităților.

```

1 var n,s,nv,d,nr,i,j,p,t,st:integer;
2   a:array[1..24] of integer; v:array[1..4096] of integer;
3
4 function caut_binar(x:integer):integer;
5 var st,dr,mij:integer;
6 begin
7   st:=1;dr:=nv+1;
8   while st<dr do begin
9     mij:=(st+dr+1) div 2;
10    if v[mij]>x then dr:=mij-1 else st:=mij;
11  end; caut_binar:=st;
12 end;

```



```

13 begin
14 assign(input, 'fotbal.in'); reset(input); d:=32000; read(n);
15 for i:=1 to n do begin read(a[i]); inc(st, a[i]); end;
16 for i:=0 to 1 shl (n div 2)-1 do begin
17   s:=0;
18   for j:=0 to n div 2-1 do
19     if i and (1 shl j)>0 then inc(s, a[j+1]);
20   inc(nv); v[nv]:=s;
21 end;
22 for i:=1 to n do
23   for j:=i+1 to n do
24     if v[i]>v[j] then begin
25       t:=v[i]; v[i]:=v[j]; v[j]:=t;
26     end;
27 for i:=0 to (1 shl (n-n div 2))-1 do begin
28   s:=0;
29   for j:=0 to n-n div 2-1 do
30     if i and (1 shl j)>0 then inc(s, a[n div 2+1+j]);
31   p:=caut_binar(st div 2-s);
32   t:=abs(2*s+2*v[p]-st);
33   if d>t then begin d:=t; Nr:=1; end
34   else
35     if d=t then inc(nr);
36 end;
37 assign(output, 'fotbal.out'); rewrite(output);
38 writeln(d, ' ', nr);
39 end.

```

```

1 #include <stdio.h>
2 int n, a[24], S, V[4096], nv, D=32000, Nr=0;
3
4 int abs(int x) { return x<0?-x:x; }
5
6 int caut_binar(int x) {
7   int st, dr, mij;
8   for (st=0, dr=nv-1; st<dr;) {
9     mij=(st+dr+1)/2;
10    if (V[mij]>x) dr=mij-1; else st=mij;
11  }
12  return st;
13 }
14
15 void main() {
16   int i, j, s, p, t;
17   freopen("fotbal.in", "r", stdin);
18   scanf("%d", &n);
19   for (i=0; i<n; i++) scanf("%d", &a[i]);
20   for (i=0; i<(1<<(n/2)); i++) {
21     for (s=j=0; j<n/2; j++)
22       if (i&(1<<j)) s+=a[j];
23     V[nv++]=s;
24   }
25   for (i=0; i<nv; i++)
26     for (j=i+1; j<nv; j++)
27       if (V[i]>V[j]) t=V[i], V[i]=V[j], V[j]=t;

```

```

27 for (i=0; i<(1<<(n-n/2)); i++) {
28   for (s=j=0; j<n-n/2; j++)
29     if (i&(1<<j)) s+=a[n/2+j];
30   p=caut_binar(S/2-s);
31   t=abs(2*s+2*V[p]-S);
32   if (D>t) D=t, Nr=1; else if (D==t) Nr++;
33   freopen("fotbal.out", "w", stdout);
34   printf("%d %d\n", D, Nr);
35 }

```

9. (Găuri - **) Se consideră o matrice cu $m \leq 100$ linii și $n \leq 100$ coloane, ale cărei elemente pot avea fie valoarea 0, fie valoarea 1. O zonă este o mulțime de elemente care au valoarea 0. Două elemente învecinate (pe verticală sau orizontală), care au ambele valoarea 0, vor face parte din aceeași zonă. O gaură este o zonă care nu conține elemente aflate pe prima linie, pe ultima linie, pe prima coloană sau pe ultima coloană a matricei. În elementele din găurile matricei trebuie scrise numere. Toate elementele din aceeași gaură trebuie să aibă aceeași valoare. Va trebui să determinați cel mai mic număr care poate reprezenta suma elementelor din oricare dintre găuri.

Fișierul de intrare *gauri.in* conține pe prima linie două numere naturale m și n separate printr-un spațiu care reprezintă dimensiunile matricei. Pe fiecare dintre următoarele m linii se află un șir de n numere care pot fi 0 sau 1 și care nu sunt separate prin spații. Aceste numere reprezintă elementele matricei.

Fișierul de ieșire *gauri.out* va conține cel mai mic număr care poate reprezenta suma elementelor din oricare dintre găurile matricei. Va exista întotdeauna cel puțin o gaură.

Exemplu:	gauri.in	gauri.out
	10 10	6
	1110010101	
	1010111111	
	1010110001	
	1111110011	
	1000010111	
	1011111111	
	1011111111	
	1000000000	
	1111111111	

Soluție:

Problema poate fi foarte ușor rezolvată prin folosirea unui simplu algoritm recursiv de umplere. Cu ajutorul acestuia se vor putea determina toate zonele. După determinarea zonelor, se vor elimina cele care nu sunt găuri (conțin cel puțin un element care se află pe prima linie, prima coloană, ultima linie sau ultima coloană) și alegem cea mai mare zonă rămasă. Se pot eticheta elementele care formează zonele folosind numere întregi distincte. După identificarea unei zone, se va păstra numărul de elemente care o formează (dimensiunea sa) într-un vector. Se va parcurge apoi prima linie, prima coloană, ultima linie și ultima coloană și se vor

determina toate etichetele care apar. Pentru fiecare etichetă determinată, se va "elimina" zona respectivă prin setarea dimensiunii sale la 0. În final, se va determina cel mai mic multiplu comun al șirului care reprezintă dimensiunile zonelor.

```

1 var n,m,t,i,j,rez:integer;
2 u:array[1..100,1..100] of integer; nr:array[1..10000] of integer;
3 a:array[1..100] of string[100]; ok:array[0..10000] of boolean;
4
5 procedure fill(x,y,t:integer);
6 begin
7   if u[x,y]<>0 then exit;
8   u[x,y]:=t; inc(nr[t]);
9   if (x+1<=n) and (a[x+1,y]='0') then fill(x+1,y,t);
10  if (x-1>0) and (a[x-1,y]='0') then fill(x-1,y,t);
11  if (y+1<=m) and (a[x,y+1]='0') then fill(x,y+1,t);
12  if (y-1>0) and (a[x,y-1]='0') then fill(x,y-1,t);
13 end;
14
15 function cmmdc(a,b:integer):integer;
16 begin if b=0 then cmmdc:=a else cmmdc:=cmmdc(b,a mod b) end;
17
18 assign(input,'gauri.in'); reset(input);
19 readln(n,m);
20 for i:=1 to n do readln(a[i]);
21 for i:=1 to n do
22   for j:=1 to m do
23     if (u[i,j]=0) and (a[i,j]='0') then begin inc(t); fill(i,j,t);
24   end;
25 for i:=1 to n do begin ok[u[i,1]]:=true; ok[u[i,m]]:=true; end;
26 for i:=1 to m do begin ok[u[1,i]]:=true; ok[u[n,i]]:=true; end;
27 rez:=1;
28 for i:=1 to t do
29   if not ok[i] then rez:=rez*nr[i] div cmmdc(rez, nr[i]);
30 assign(output,'gauri.out'); rewrite(output);
31 writeln(rez);

```

```

1 #include <stdio.h>
2 int n,m,t,rez=1,u[100][100],nr[10000]; char a[102][102],ok[10000];
3
4 void fill(int x,int y,int t) {
5   if (u[x][y]) return;
6   u[x][y]=t; nr[t]++;
7   if (x+1<n&& a[x+1][y]=='0') fill(x+1,y,t);
8   if (x-1>0&& a[x-1][y]=='0') fill(x-1,y,t);
9   if (y+1<m&& a[x][y+1]=='0') fill(x,y+1,t);
10  if (y-1>0&& a[x][y-1]=='0') fill(x,y-1,t);
11 }
12
13 int cmmdc(int a,int b) {
14   if (b==0) return a;
15   else return cmmdc(b,a%b);
16 }

```

```

17 void main() {
18   int i,j;
19   freopen("gauri.in","r",stdin);
20   scanf("%d %d\n",&n,&m);
21   for (i=0;i<n;i++) fgets(a[i],m+2,stdin);
22   for (i=0;i<n;i++)
23     for (j=0;j<m;j++)
24       if (u[i][j]&&a[i][j]=='0') fill(i,j,++t);
25   for (i=0;i<n;i++) ok[u[i][0]]=ok[u[i][m-1]]=1;
26   for (i=0;i<m;i++) ok[u[0][i]]=ok[u[n-1][i]]=1;
27   for (i=1;i<=t;i++) if (!ok[i]) rez=rez*nr[i]/cmmdc(rez, nr[i]);
28   freopen("gauri.out","w",stdout);
29   printf("%d\n",rez);
30 }

```

10. (Lămpi - ***) Într-un castel sunt $N \leq 10.000$ lămpi numerotate de la 1 la N . Fiecare lampă poate fi aprinsă ori stinsă. La fiecare secundă, lampa cu numărul i își schimbă starea dacă lampa cu numărul $i+1$ este aprinsă, cu excepția lămpii N care își schimbă starea dacă lampa cu numărul 1 este aprinsă. Dându-se starea inițială a lămpilor, să se determine starea lămpilor după $M \leq 1.000.000.000$ secunde.

Fișierul de intrare *lampi.in* conține, pe prima linie, numerele N și M . Următoarele N linii conțin starea inițială a lămpilor, numărul 0 semnificând că lampa este stinsă, iar 1 că este aprinsă.

Fișierul de ieșire *lampi.out* va conține N linii reprezentând starea lămpilor după M secunde.

Exemplu:	lampi.in	lampi.out
3 1	0	
0	1	
0	1	
1		

Soluție:

Se notează cu $A[i, T]$ starea lămpii i la timpul T . Se presupune că pentru k este adevărat că $A[i, T+2^k] = A[i, T] \text{ XOR } A[i+2^k, T]$ (se consideră că numerele sunt în cerc, adică se iau resturile indicilor la N) și se demonstrează pentru $k+1$.

$$A[i, T+2^{k+1}] = A[i, T+2^k+2^k] = A[i, T+2^k] \text{ XOR } A[i+2^k, T+2^k] =$$

$$(A[i, T] \text{ XOR } A[i+2^k, T]) \text{ XOR } (A[i+2^k, T] \text{ XOR } A[i+2^{k+1}, T]) =$$

$$A[i, T] \text{ XOR } A[i+2^{k+1}, T]$$

Așadar pentru a afla starea după M secunde se fac repetat salturi cu puteri ale lui 2, cât mai mari.

```

1 var n,m,i,p:longint; a,b:array[0..9999] of byte;
2
3 procedure gen(x:longint);
4 var i:longint;
5 begin
6   for i:=0 to n-1 do b[i]:=a[i] xor a[(i+x) mod n];
7   for i:=0 to n-1 do a[i]:=b[i];
8 end;

```

```

9 begin
10 assign(input, 'lampi.in'); reset(input); readln(n,m);
11 for i:=0 to n-1 do read(a[i]);
12 while m>0 do begin
13   p:=1; while p<=m do p:=p*2; p:=p div 2;
14   gen(p); m:=m-p;
15 end;
16 assign(output, 'lampi.out'); rewrite(output);
17 for i:=0 to n-1 do writeln(a[i]);
18 end.

```

```

1 #include <stdio.h>
2 long n,m; char a[10000],b[10000];
3 void gen(long x) {
4   long i;
5   for (i=0;i<n;i++) b[i]=a[i]^a[(i+x)%n];
6   for (i=0;i<n;i++) a[i]=b[i];
7 }
8
9 void main() {
10  long i,p;
11  freopen("lampi.in","r",stdin); scanf("%ld %ld",&n,&m);
12  for (i=0;i<n;i++) scanf("%d",&a[i]);
13  while (m>0) {
14    for (p=1;p<=m;p*=2);p/=2;
15    gen(p); m-=p;
16  }
17  freopen("lampi.out","w",stdout);
18  for (i=0;i<n;i++) printf("%d\n",a[i]);
19 }

```

11. (Secvența - **) Gheorghe a dat peste o nouă problemă de informatică la care are nevoie de un pic de ajutor! Dându-se un șir de $N \leq 5.000$ numere naturale, aflați lungimea minimă a unei subsecvențe care conține un subșir strict crescător, iar acest subșir conține toate numere din șirul inițial o singură dată. Dacă nu există o astfel de secvență, răspunsul va fi -1.

Pe prima linie a fișierului de intrare se găsește N , lungimea șirului. Pe a doua linie se găsesc N numere întregi, șirul propriu zis. Elementele șirului sunt numere întregi din intervalul $[0, 2.000.000.000]$.

Pe prima linie a fișierului de ieșire se găsește numărul cerut.

Exemplu: secv.in secv.out

8
2 1 3 2 1 3 4 5

7

(http://infoarena.devnet.ro)

Soluție:

Subșirul trebuie să conțină toate elementele din șirul original în ordine crescătoare așa că primul pas este de a forma acest subșir C . Având acest subșir, se parcurge vectorul inițial pentru a găsi poziția de start a noii subsecvențe. După ce s-a găsit o posibilă poziție de start s , se încearcă să găsim subșirul C având ca poziție de start s . Din toate aceste subsecvențe se alege pe aceea cu lungimea minimă.

```

1 var n,m,rez,i,j,k,x:longint;
2   v,t,c:array[1..5000] of longint;
3 begin
4   assign(input, 'secv.in'); reset(input);
5   readln(n); rez:=1000000000;
6   for i:=1 to n do begin read(v[i]); t[i]:=v[i]; end;
7   for i:=1 to n do
8     for j:=i+1 to n do
9       if t[i]>t[j] then begin x:=t[i]; t[i]:=t[j]; t[j]:=x; end;
10    c[1]:=t[1]; m:=1;
11    for i:=2 to n do
12      if t[i]<>t[i-1] then begin inc(m); c[m]:=t[i]; end;
13    for i:=n downto 1 do begin
14      if v[i]=c[m] then begin
15        k:=i;
16        for j:=m-1 downto 1 do begin
17          while (k>0) and (v[k]<>c[j]) do dec(k);
18          if (k<=0) then break; end;
19          if (k>0) and (rez>i-k+1) then rez:=i-k+1;
20        end;
21      end;
22    assign(output, 'secv.out'); rewrite(output);
23    if rez=1000000000 then writeln(-1) else writeln(rez);
24  end.

```

```

1 #include <stdio.h>
2 long n,m,v[5000],t[5000],c[5000],rez=1000000000;
3 void main() {
4   long i,j,k,x;
5   freopen("secv.in","r",stdin);
6   scanf("%ld",&n);
7   for (i=0;i<n;i++) { scanf("%ld",&v[i]); t[i]=v[i]; }
8   for (i=0;i<n;i++)
9     for (j=i+1;j<n;j++)
10      if (t[i]>t[j]) x=t[i], t[i]=t[j], t[j]=x;
11   c[0]=t[0]; m=1;
12   for (i=1;i<n;i++)
13     if (t[i]!=t[i-1]) c[m++]=t[i];
14   for (i=n-1;i>=0;i--) {
15     if (v[i]=c[m-1]) {
16       for (k=i,j=m-2;j>=0;j--) {
17         while (k>=0 && v[k]!=c[j]) k--;
18         if (k<0) break; }
19       if (j== -1 && rez>i-k+1) rez=i-k+1;
20     } }
21   freopen("secv.out","w",stdout);
22   printf("%ld\n",rez==1000000000?-1:rez);
23 }

```

12. (Farfurii - ***) În fiecare zi, Zăhărel este obligat de Eugenia să spele farfuriile și tacâmurile după fiecare masă. După ce le spală, el trebuie să le aranjeze pe două rafturi, farfuriile pe primul și tacâmurile pe al doilea... dar nu oricum! El are $N \leq 10.000$ farfurii de mărimi distincte, cuprinse între 1 și N și $K \leq N*(N-1)/2$

tacâmuri identice. Pentru fiecare pereche de farfurii așezate în raft astfel încât farfuria de mărime mai mare, dintre cele două, apare înaintea farfuriei de mărime mai mică, Zăhărel pune un tacâm pe rândul al doilea.

Ajutați-l pe Zăhărel să așeze toate farfuriile pe primul raft astfel încât să pună toate tacâmurile pe al doilea raft. Dintre toate așezările posibile, determinați-o pe aceea minim lexicografică din punct de vedere al mărimilor. O așezare (A_1, A_2, \dots, A_N) este mai mică din punct de vedere lexicografic decât o altă așezare (B_1, B_2, \dots, B_N) dacă există o poziție p astfel încât $A_p < B_p$ și $A_1 = B_1, A_2 = B_2, \dots, A_{p-1} = B_{p-1}$.

Pe prima linie din fișierul de intrare *farfurii.in* se găsesc numerele naturale N și K . Pe prima linie din fișierul de ieșire *farfurii.out* se vor găsi N numere distincte între 1 și N reprezentând mărimile farfuriilor, afișate în ordinea în care au fost așezate pe raft.

Exemplu:

<i>farfurii.in</i>		<i>farfurii.out</i>
7 8		1 2 5 7 6 4 3

(<http://infoarena.devnet.ro>)

Soluție:

O rezolvare simplă se bazează pe următoarea observație: "O permutare de lungime i are cel mult $i*(i-1)/2$ inversiuni când numerele sunt în ordine descrescătoare".

Astfel, dacă K e de forma $M*(M-1)/2$ permutarea minim lexicografică cu K inversiuni va fi 1, 2, 3, ..., $N-M$, N , $N-1$, $N-2$, ..., $N-M+1$. Cele K inversiuni apar în ultimele M elemente. Dacă în această permutare se mută un element $N-x$ imediat înaintea lui N numărul de inversiuni scade cu x . Astfel, dacă $K > M*(M-1)/2$ se construiește permutarea: 1, 2, 3, ..., $N-M-1$, N , $N-1$, $N-2$, ..., $N-M$ (care are $(M+1)*M/2$ inversiuni) și se mută elementul $N-((M+1)*M/2-K)$ imediat înaintea lui N , astfel se scade numărul de inversiuni la K . Este evident că permutarea astfel construită este minim lexicografică.

```

1 var n,k,i,m,p:longint;
2 begin
3   assign(input,'farfurii.in'); reset(input);
4   readln(n,k);
5   assign(output,'farfurii.out');
6   rewrite(output);
7   p:=1;
8   while p<n do p:=p*2;
9   m:=0;
10  while p>0 do begin
11    if (m+p<=n) and ((m+p)*(m+p-1)<=2*k) then m:=m+p;
12    p:=p div 2;
13  end;
14  k:=k-m*(m-1) div 2;
15  if k=0 then begin
16    for i:=1 to n-m do write(i,' ');
17    for i:=n downto n-m+1 do write(i,' ');
18    writeln;
19  end

```

```

20 else begin
21   k:=m-k; inc(m);
22   for i:=1 to n-m do write(i,' ');
23   write(n-k,' ');
24   for i:=n downto n-k+1 do write(i,' ');
25   for i:=n-k-1 downto n-m+1 do write(i,' ');
26   writeln;
27 end
28 end.

```

```

1 #include <stdio.h>
2 long n,k;
3 void main() {
4   long i,m,p;
5   freopen("farfurii.in","r",stdin);
6   freopen("farfurii.out","w",stdout);
7   scanf("%ld %ld",&n,&k);
8   for (p=1;p<n;p*=2);
9   for (m=0;p>0;p/=2)
10    if (m+p<=n && (m+p)*(m+p-1)<=2*k) m+=p;
11   k -= m*(m-1)/2;
12   if (!k) {
13     for (i=1;i<=n-m;i++) printf("%ld ",i);
14     for (i=n;i>n-m;i--) printf("%ld ",i);
15     putchar('\n'); }
16   else {
17     k=m-k; m++;
18     for (i=1;i<=n-m;i++) printf("%ld ",i);
19     printf("%ld ",n-k);
20     for (i=n;i>n-k;i--) printf("%ld ",i);
21     for (i=n-k-1;i>n-m;i--) printf("%ld ",i);
22     putchar('\n');
23   }
24 }

```

13. (*Perechi* - **) Se dă un număr întreg strict pozitiv $N \leq 2^{31}$. Trebuie să determinați câte perechi de numere întregi strict pozitive au cel mai mic multiplu comun egal cu N . Perechile (a,b) și (b,a) se consideră identice.

În fișierul de intrare *perechi.in* se afla numărul N .

În fișierul de ieșire *perechi.out* veți afișa numărul determinat.

Exemplu:

<i>perechi.in</i>		<i>perechi.out</i>
12		8

(<http://infoarena.devnet.ro>)

Soluție:

Se va factoriza numărul N ; din definiția celui mai mare multiplu comun a două numere, se iau toate numerele prime din factorizare la puterea cea mai mare dintre cele două la care apar. Astfel, pentru fiecare număr prin din N , dacă acesta apare la exponentul e se consideră odată că exponentul primului număr este e , iar al celuiilalt $<e$, odată că primui număr are exponentul $<e$ și al doilea e , și odată ca ambii sunt e ; astfel se înmulțește rezultatul cu 2^{*e+1} . Fiindcă astfel se număra și perechile (a,b) și (b,a) , rezultatul total se crește cu o unitate și se împarte la 2.

```

1 var n,i,j,rez:longint;
2 begin
3   assign(input,'perechi.in'); reset(input);
4   read(n);
5   i:=2;
6   rez:=1;
7   while i<=n do begin
8     j:=0; while n mod i=0 do begin n:=n div i; inc(j); end;
9     if j>0 then rez:=rez*(2*j+1);
10    inc(i);
11  end;
12  if n>1 then rez:=rez*3;
13  assign(output,'perechi.out'); rewrite(output);
14  writeln((rez+1) div 2);
15 end.

```

```

1 #include <stdio.h>
2 long n, rez=1;
3 void main() {
4   long i, j;
5   freopen("perechi.in", "r", stdin);
6   scanf("%d", &n);
7   for (i=2; i<=n; i++) {
8     for (j=0; n%i==0; n/=i, j++);
9     if (j>0) rez*= (2*j+1);
10  }
11  if (n>1) rez*=3;
12  freopen("perechi.out", "w", stdout);
13  printf("%ld\n", (rez+1)/2);
14 }

```

14. (Flip - ***) Gigel a descoperit un nou joc pe care l-a numit „Flip”. Acesta se joacă pe o tablă dreptunghiulară de dimensiuni $N \times M$ ($1 \leq N, M \leq 16$) care conține numere întregi. Fiecare linie și fiecare coloană are un comutator care schimbă starea tuturor elementelor de pe acea linie sau coloană, înmulțindu-le cu -1. Scopul jocului este ca, pentru o configurație dată a tablei de joc, să se acționeze asupra liniilor și coloanelor astfel încât să se obțină o tablă cu suma elementelor cât mai mare.

Prima linie a fișierului *flip.in* conține două numere întregi N și M , separate prin câte un spațiu, care reprezintă dimensiunea tablei. Următoarele N linii conțin câte M numere întregi separate prin câte un spațiu care descriu configurația tablei de joc. Tabla de joc conține numere întregi din intervalul $[-1.000.000, 1.000.000]$.

Prima linie a fișierului *flip.out* conține un număr care va reprezenta suma maximă pe care Gigel o poate obține comutând liniile și coloanele tablei de joc.

Exemplu: *flip.in*

```

5 3
4 -2 2
3 -1 5
2 0 -3
4 1 -3
5 -3 2

```

28

flip.out

(<http://infoarena.devnet.ro>)

Soluție:

Soluția constă în a genera acțiunile de comutatoare de pe prima linie a tabloului. În acest mod, pe fiecare linie se va alege dacă se comută sau nu linia, în funcție de modul în care se obține sumă mai mare. Dintre toate aceste variante de acționări, se va păstra aceea cu suma maximă.

```

1 var n,m,i,j,k,s,t,smax:longint;
2   a:array[1..16,1..16] of longint;
3 begin
4   assign(input,'flip.in'); reset(input); readln(n,m);
5   for i:=1 to n do
6     for j:=1 to m do read(a[i,j]);
7   for i:=0 to (1 shl m)-1 do begin
8     s:=0;
9     for k:=1 to n do begin
10      t:=0;
11      for j:=1 to m do
12        if i and (1 shl (j-1))>0 then inc(t,-a[k,j]) else inc(t,a[k,j]);
13      if t<-t then inc(s,-t) else inc(s,t);
14    end;
15    if smax<s then smax:=s;
16  end;
17  assign(output,'flip.out'); rewrite(output);
18  writeln(smax);
19 end.

```

```

1 #include <stdio.h>
2 long n,m,a[16][16],smax;
3 void main() {
4   long i,j,k,s,t;
5   freopen("flip.in", "r", stdin);
6   scanf("%lld %lld", &n, &m);
7   for (i=0; i<n; i++)
8     for (j=0; j<m; j++) scanf("%lld", &a[i][j]);
9   for (i=0; i<(1<m); i++) {
10    s=0;
11    for (k=0; k<n; k++) {
12      for (t=j=0; j<m; j++)
13        if (i & (1<j)) t+=-a[k][j]; else t+=a[k][j];
14      s+=t<-t ? -t : t;
15    }
16    if (smax<s) smax=s;
17  }
18  freopen("flip.out", "w", stdout); printf("%lld\n", smax);
19 }

```

15. (Numere Prime - ***) Se dă un număr natural $N \leq 100.000$. Să se genereze toate numerele prime mai mici sau egale cu N .

Prima linie a fișierului *nrprime.in* conține numărul întreg N .

Prima linie a fișierului *nrprime.out* va conține numerele prime mai mici sau egale cu N , câte unul pe o linie.

Exemplu:	nrprime.in		nrprime.out
10		2	
		3	
		5	
		7	

Soluție:

O soluție clasică este folosirea ciurului lui Eratostene. În continuare se va prezenta o soluție recursivă mai rapidă decât ciurul lui Eratostene. Se va construi o procedură care va determina pentru un număr n dat, un vector $A[i]$ care va fi 0 în caz că i este prim sau cel mai mare factor prim al lui i , în caz contrar. Aceasta se va apela recursiv pentru $n/2$ și va determina folosind rezultatele anterioare vectorul A . Datorită dimensiunii datelor se vor folosi compilatoarele FreePascal sau GCC.

```

1 var n,i:integer; a,urm:array[0..100000] of integer;
2
3 procedure gen(n:integer);
4 var i,j:integer;
5 begin
6   if n<=1 then exit;
7   for i:=0 to n do urm[i]:=0;
8   if n=2 then begin urm[0]:=2; urm[2]:=-1; exit; end;
9   if n=3 then begin urm[0]:=2; urm[2]:=3; urm[3]:=-1; exit; end;
10  gen(n div 2);
11  a[1]:=1;
12  for i:=2 to n do begin
13    j:=a[i];
14    while (j<>-1)and(i*j<=n) do begin
15      a[i*j]:=j;
16      j:=urm[j];
17    end;
18  end;
19  j:=0;
20  for i:=1 to n do
21    if a[i]=0 then begin urm[j]:=i; j:=i; end;
22  urm[j]:=-1;
23 end;
24
25 begin
26   assign(input,'nrprime.in'); reset(input);
27   read(n); gen(n);
28   i:=urm[0];
29   assign(output,'nrprime.out'); rewrite(output);
30   while i<>-1 do begin
31     writeln(i); i:=urm[i];
32   end;
33 end.
```

```

1 #include <stdio.h>
2 int n, urm[100001], a[100001];
3 void gen(int n) {
4   int i,j;
5   if (n<=1) return;
```

```

6   for (i=0;i<=n;i++) urm[i]=0;
7   if (n==2) { urm[0]=2; urm[2]=-1; return; }
8   if (n==3) { urm[0]=2; urm[2]=3; urm[3]=-1; return; }
9   gen(n/2);
10  for (a[1]=1,i=2;i<=n;i++) {
11    j=a[i];
12    while (j!=-1&&i*j<=n) { a[i*j]=j; j=urm[j]; }
13  }
14  for (j=0,i=1;i<=n;i++)
15    if (!a[i]) urm[j]=i,j=i;
16  urm[j]=-1; }
17
18 void main() {
19   int i;
20   freopen("nrprime.in","r",stdin); scanf("%d",&n);
21   gen(n);
22   freopen("nrprime.out","w",stdout);
23   for (i=urm[0];i!=-1;i=urm[i])
24     printf("%d\n",i);
25 }
```

2.3.2 Probleme propuse

1. (Cuvinte - ****) Doi prieteni, Marius și Andrei, s-au gândit la un joc. Marius scrie, pe o foaie, un șir de $N \leq 200$ numere. Sub fiecare număr, el scrie câte o literă: sub primul număr litera A, sub al doilea număr litera B, și tot așa, în ordine lexicografică. Marius și Andrei folosesc un alfabet cu câteva mii de litere, cunoscut numai de ei și care începe cu literele de la A la Z. Literele sunt deci folosite ca indici pentru numerele din șir. Andrei caută apoi toate "cuvintele" posibile care respectă următoarele condiții: un cuvânt reprezintă un șir de litere, ordonat lexicografic; numerele din șir, corespunzătoare literelor dintr-un cuvânt și scrise în ordinea dată de acestea, sunt în ordine strict crescătoare. De exemplu, pentru șirul 2 1 3 5 4, scriind dedesubt literele ABCDE, câteva dintre cuvintele valide sunt AC, ACD, ACE, AD, BE etc., dar AB, ED sau BDE nu sunt cuvinte valide. Apoi, Andrei alege dintre aceste cuvinte, pe cele de lungime maximă și le scrie în ordine lexicografică. Pentru exemplul de mai sus, acestea sunt ACD, ACE, BCD și BCE. Dintre aceste cuvinte de lungime maximă, el i-l spune lui Marius pe al K -lea ($K \leq 2.000.000.000$) în ordine lexicografică. Dacă Andrei spune corect (și repede) cuvântul acesta, el câștigă jocul și îl pierde în caz contrar. Andrei câștiga întotdeauna și era foarte mândru de el, până într-o bună zi, când Marius i-a scris un șir de 67 de numere... I-ar fi fost cam greu lui Andrei să facă tot ce făcea de obicei, așa că s-a gândit să scrie un program care să-i dea direct rezultatul. Scrieți un program care determină cuvântul cerut și îl ajută pe Andrei să câștige jocul. Fișierul de intrare *cuvinte.in* conține, pe prima linie, două numere N și K , separate printr-un spațiu, reprezentând numărul de numere din șir, respectiv numărul de ordine al cuvântului cerut. Pe a doua linie se află N numere întregi separate prin câte un spațiu, numerele scrise de Marius pe foaie. Numerele din șir sunt întregi, cuprinse între 0 și 10000 inclusiv.

În fișierul *cuvinte.out* se va scrie cuvântul cerut. Deoarece nu se cunosc literele care urmează după Z în alfabetul celor doi, în locul literelor cuvântului se vor scrie numerele de ordine ale acestora în alfabet, separate prin câte un spațiu. Astfel, de exemplu, cuvântul ACZ ar fi scris ca 1 3 26. Deci, în fișier, se vor scrie, separate printr-un spațiu, numere întregi care reprezintă numerele de ordine în alfabet ale literelor celui mai lung cuvânt cu numărul de ordine K , format după regulile din enunț.

Exemplu:

<i>cuvinte.in</i>		<i>cuvinte.out</i>
5 3		2 3 4
2 1 3 5 4		

("Stelele informaticii" 2003, București)

2. (Aliniere - ***) Un grup de n războinici elfi stau aliniați în linie dreaptă în fața conducătorului lor. Acesta nu e mulțumit de ceea ce vede; războinicii nu sunt așezați în ordinea înălțimii. Căpitanul cere câtorva războinici să iasă din rând, astfel ca cei rămași, fără a-și schimba locurile, doar apropiindu-se unul de altul (pentru a nu rămâne spații mari între ei), să formeze un șir în care fiecare războinic vede privind de-a lungul șirului, cel puțin una din extremități (stânga sau dreapta). Un războinic vede o extremitate dacă între el și capătul respectiv nu există un alt războinic cu înălțimea mai mare sau egală cu a lui. Așadar, trebuie determinat numărul minim de războinici care trebuie să părăsească formația, astfel încât șirul rămas să respecte condiția impusă de conducător.

Prima linie a fișierului de intrare *aliniere.in* conține numărul n al războinicilor din șir. Numărul războinicilor din șir este cuprins între 2 și 1000. Fiecare dintre următoarele n linii va conține înălțimea unui războinic din șir. Aceste linii vor respecta ordinea amplasării inițiale a războinicilor; înălțimile războinicilor sunt numere reale, cu cel mult 5 zecimale, cuprinse între 1 și 10.

Fișierul de ieșire *aliniere.out* trebuie să conțină, pe prima linie, numărul k al războinicilor care trebuie să părăsească șirul. Pe fiecare dintre următoarele k linii se va afla numărul de ordine al unui războinic care trebuie să părăsească șirul. Aceste numere vor fi scrise în ordine crescătoare. Dacă există mai multe soluții va putea fi aleasă oricare dintre ele.

Exemplu:

<i>aliniere.in</i>		<i>aliniere.out</i>
8		4
1.86		1
1.86		3
1.30621		7
2		8
1.4		
1		
1.97		
2.2		

3. (Munte - ***) Un număr are aspect de munte dacă există o poziție k , astfel încât $c_1 \leq c_2 \leq \dots \leq c_k$ și $c_k \geq c_{k+1} \geq \dots \geq c_n$, unde c_1, \dots, c_n sunt cifrele numărului. Dându-se un număr, să se determine numărul minim de cifre care trebuie eliminate astfel încât numărul rămas să aibă aspect de munte.

Fișierul de intrare *munte.in* conține, o singură linie, pe care se află numărul dat. Numărul va conține cel mult 5000 de cifre. Cifrele numărului sunt nenule. Fișierul de ieșire *munte.out* va conține, o singură linie, pe care se va afla numărul minim al cifrelor care trebuie eliminate.

Exemplu:

<i>munte.in</i>		<i>munte.out</i>
248793		1

4. (Cărți - ****) Lui Vasilică îi plac la nebunie jocurile de cărți, dar, fiindcă are doar 4 ani, pierde de fiecare dată când joacă cu prietenii săi mai mari. Până și aranjarea cărților în mână e o problemă pentru el. Când primește o mână la cărți, Vasilică trebuie să aranjeze cărțile pe grupe astfel încât cărțile din aceeași grupă să aibă aceeași culoare, iar apoi trebuie să ordoneze cărțile din aceeași grupă după valoare, astfel încât cartea cu cea mai mică valoare să fie plasată cât mai la stânga. Și, ce-i mai rău, el trebuie să țină permanent toate cărțile în mână. Vasilică trebuie să aranjeze cărțile cât mai rapid posibil, adică făcând un număr minim de mutări. O mutare constă în plasarea unei cărți pe o altă poziție. Scrieți un program care să determine numărul minim de mutări necesare pentru ca Vasilică să aranjeze cărțile din mână.

Fișierul de intrare *carti.in* conține, pe prima linie, două numere naturale $C \leq 6$ (reprezentând numărul de culori din jocul de cărți) și $N \leq 1.000$ (reprezentând numărul de cărți de aceeași culoare), separate printr-un spațiu. Fiecare dintre următoarele $C \cdot N$ linii conține o pereche de numere naturale x și y , separate printr-un spațiu, reprezentând culoarea (x) și valoarea (y) cărților pe care le-a tras Vasilică, în ordinea extragerii. Oricare două cărți extrase de Vasilică sunt distincte. Fișierul de ieșire *carti.out* conține o singură linie, pe care se află numărul minim de mutări pe care trebuie să le facă Vasilică pentru a aranja toate cărțile.

Exemplu:

<i>carti.in</i>		<i>carti.out</i>
3 2		2
3 2		
2 2		
1 1		
3 1		
2 1		
1 2		

(<http://campion.edu.ro>)

5. (Grupuri - **) Se consideră N numere întregi care trebuie repartizate în p grupuri. Grupurile sunt identificate prin numere naturale cuprinse între 1 și p . Repartizarea în grupuri trebuie să se realizeze astfel încât suma numerelor din oricare grup i să fie divizibilă cu numărul total de numere care fac parte din grupurile identificate prin numere cuprinse între 1 și p . Prima linie a fișierului de intrare *grupuri.in* conține numărul $N \leq 1.000$ al valorilor care trebuie repartizate în grupuri. Cea de-a doua linie va conține cele N numere (cuprinse între 1 și 1.000), separate prin câte un spațiu.

Fișierul de ieșire *grupuri.out* trebuie să conțină un număr de linii egal cu numărul grupurilor. Prima linie corespunde primului grup, a doua linie celui de-al doilea grup etc. Linia corespunzătoare unui grup va conține numerele care fac parte din grupul respectiv, separate prin câte un spațiu. Numărul grupurilor nu este cunoscut; el va trebui determinat de program. Fiecare grup va conține cel puțin un număr; dacă există mai multe soluții va fi generată doar una dintre ele; va exista întotdeauna cel puțin o soluție.

Exemplu: *grupuri.in* *grupuri.out*

6	2 4
4 10 3 9 2 3	3 9
	10
	3

6. (Cifre - ***) Se consideră un număr natural n și k cifre distincte între care se află, cu siguranță, și cifra 0. Să se determine un multiplu al lui n care conține numai cifrele date, fiecare dintre ele fiind prezentă cel puțin o dată în acest multiplu. Prima linie a fișierului de intrare *cifre.in* conține valoarea $n \leq 1.000$. Cea de-a doua linie a fișierului conține cifrele, neseperate prin spații. Fișierul de ieșire *cifre.out* va conține o singură linie pe care se va afla multiplul determinat.

Exemplu: *cifre.in* *cifre.out*

11	1210
012	

7. (Divizori - ****) Pentru un număr $n \leq 1.000$ dat, să determine cel mai mic număr natural care are exact n divizori. Fișierul de intrare *divizori.in* conține, pe prima linie, numărul n al divizorilor pe care trebuie să îi aibă numărul căutat. Fișierul de ieșire *divizori.out* trebuie să conțină o singură linie, pe care se va afla cel mai mic număr natural care are exact n divizori.

Exemplu: *divizori.in* *divizori.out*

6	12
---	----

8. (Suma - ***) Pe o masă se află $n \leq 100$ cartonașe; pe fiecare dintre cartonașe este scris un număr natural cuprins între 0 și 1.000. De pe masă pot fi alese oricâte cartonașe. După alegere se calculează suma numerelor de pe cartonașele alese. Va trebui să determinați numărul sumelor distincte care pot fi obținute prin astfel de alegeri.

Prima linie a fișierului de intrare *sume.in* conține numărul n al cartonașelor. Cea de-a doua linie a fișierului conține cele n numere scrise pe cartonașe, separate prin spații.

Fișierul de ieșire *sume.out* va conține numărul sumelor distincte care pot fi obținute.

Exemplu: *sume.in* *sume.out*

5	11
0 1 2 3 4	

9. (Puteri - *) Elfilor le place să înmulțească de multe ori un număr cu el însuși. Este ceea ce noi numim acum ridicare la putere. Ei pornesc de obicei de la un număr mic și se distrează văzând cât de mare poate ajunge numărul prin înmulțiri succesive.

Fișierul de intrare *puteri.in* conține două linii. Pe prima linie se află un număr $a \leq 1.000$, iar pe cea de-a doua se află un număr $n \leq 1.000$.

Fișierul de ieșire *puteri.out* trebuie să conțină o singură linie pe care se va afla numărul obținut prin înmulțirea numărului a cu el însuși de un anumit număr (n) de ori (este ceea ce noi numim a^n).

Exemplu: *puteri.in* *puteri.out*

3	282429536481
24	

10. (Puncte - ****) Se consideră $n \leq 10.000$ puncte în plan. Să se determine numărul posibilităților de a alege trei dintre aceste puncte, astfel încât aria triunghiului determinat de acestea să fie un număr întreg.

Fișierul de intrare *puncte.in* conține, pe prima linie, numărul n al punctelor din plan. Fiecare dintre următoarele n linii va conține câte două numere, separate prin spații, reprezentând coordonatele unui punct. Coordonatele punctelor sunt numere întregi cuprinse între 0 și 1000. Aria triunghiului determinat de trei puncte coliniare este considerată a fi 0.

Fișierul de ieșire *puncte.out* va conține o singură linie pe care se va afla numărul posibilităților de a alege trei dintre puncte, astfel încât aria triunghiului determinat de acestea să fie un număr întreg.

Exemplu: *puncte.in* *puncte.out*

4	4
0 0	
0 2	
2 2	
2 0	

11. (Laser - **) În anul 2457, în provincia Etram de pe planeta Marte au avut mai multe explozii nucleare. Locuitorii planetei dețin o tehnologie cu ajutorul căreia pot împiedica răspândirea particulelor radioactive care sunt capabile să distrugă toate formele de viață de pe planetă. Tehnologia lor se bazează pe două unde de tip LASER care, transmise din același punct pe direcții diferite, distrug toate particulele radioactive care se află între ele sau le intersectează. Locuitorii planetei au ales un loc la marginea provinciei, unde au instalat un dispozitiv bazat pe tehnologia prezentată anterior. Dispozitivul este montat în punctul de coordonate (0, 0). Datorită faptului că locuitorii planetei doresc ca cele două raze să fie cât mai apropiate posibil astfel încât să nu existe pericolul răspândirii particulelor radioactive pe toată planeta, vă roagă să determinați unghiul minim pe care undele LASER îl pot forma.

Fișierul de intrare *laser.in* conține numărul $n \leq 10.000$ al punctelor în care au avut loc explozii nucleare. Pe fiecare dintre următoarele n linii se află câte două numere, separate printr-un singur spațiu, care reprezintă coordonatele punctelor în care au

Fișierul de ieșire *laser.out* trebuie să conțină o singură linie, pe care se va afla un singur număr care reprezintă valoarea unghiului minim care poate fi format de cele două raze laser ale dispozitivului. Unghiul va fi scris în grade, cu două zecimale exacte. Se garantează existența unei soluții.

Exemplu:	laser.in	laser.out
4		36.87
2 1		
1 1		
2 4		
4 3		

Fișierul de intrare, *aur.in*, conține pe prima linie un singur număr $n \leq 30.000$, care reprezintă numărul de mine care trebuie să revină fiecăruia dintre cei doi. Fiecare a i -a linie dintre următoarele $2*n$ linii conține câte 2 numere, separate între ele printr-un singur spațiu, care reprezintă coordonatele unei mine. Coordonatele minelor sunt numere întregi cuprinse între 1 și 1000.

Fișierul de ieșire *aur.out* trebuie să conțină, pe o singură linie, trei numere a , b și c separate între ele prin spații, care reprezintă coeficienții dreptei de ecuație $a \cdot x + b \cdot y + c = 0$, care are de o parte sau pe ea n mine și de cealaltă parte sau pe ea celelalte n mine. Coeficienții a , b și c vor fi scriși cu 8 zecimale exacte.

Exemplu:	aur.in	aur.out
2		3.00000000 -2.00000000 -3.00000000
1 2		
2 2		
3 1		
3 2		

13. (*Piramidă* - ***) Se consideră o piramidă triunghiulară (determinată de patru puncte necoplanare din spațiu) și alte $N \leq 100.000$ puncte din spațiu. Să se determine câte dintre aceste N puncte se află în interiorul piramidei. Un punct este considerat a fi în interiorul piramidei chiar dacă se află pe una dintre cele patru fețe, pe una dintre cele șase muchii sau este unul dintre cele patru vârfuri ale piramidei. Primele patru linii ale fișierului de intrare *piramida.in* conțin câte trei numere întregi reprezentând coordonatele vârfurilor piramidei. Următoarea linie conține numărul N al celorlalte puncte. Fiecare dintre următoarele N va conține câte trei

Fișierul de ieșire *piramida.out* trebuie să conțină o singură linie pe care se va afla numărul punctelor din interiorul piramidei.

```

Exemplu:      piramida.in      1      piramida.out
0 0 0
0 0 100
0 100 0
100 0 0
2
100 100 100
1 1 1

```

14. (*Perechi* - **) Se consideră un număr natural $n \leq 2.500$ și trebuie formate cât mai multe perechi, din care trebuie să facă parte numere cuprinse între 1 și $2n$, astfel încât suma pătratelor celor două numere din oricare dintre perechi să fie număr prim. Fiecare număr poate face parte din cel mult o pereche.

Fişierul de intrare *perechi.in* conţine o singură linie pe care se află valoarea n . Prima linie a fişierului de ieşire *perechi.out* va conţine numărul k al perechilor formate. Fiecare dintre următoarele k linii va conţine câte două numere cuprinse între 1 şi $2n$, astfel încât suma pătratelor celor două numere este număr prim. Perechile pot fi scrise în orice ordine.

<i>Exemplu:</i>	<i>perechi.in</i>			<i>perechi.out</i>
7			7	
			1 4	
			2 3	
			5 6	
			7 8	
			9 10	
			11 14	
			12 13	

15. (Ordine - ****) Se consideră $n \leq 30.000$ copii așezați în cerc și numerotați de la 1 la n în sens trigonometric. Copiii joacă următorul joc: jocul începe de la primul copil (cel al cărui număr de ordine este 1); la fiecare al i -lea pas al jocului se numără i copii în sens trigonometric și este eliminat copilul la care se ajunge; la pasul următor numărătoarea începe de la copilul care urmează după cel eliminat. Așadar, dacă numărul copiilor este suficient de mare, la primul pas este eliminat al doilea copil, la al doilea pas al patrulea, la al treilea pas al șaptelea, apoi la al patrulea pas al unsprezecelea și așa mai departe. Va trebui să determinați ordinea în care vor fi eliminați copiii.

Fișierul de intrare *ordine.in* conține pe prima linie un număr întreg n , care reprezintă numărul de copii.

Fișierul de ieșire *ordine.out* trebuie să conțină o singură linie pe care se vor afla n numere distincte cuprinse între 1 și n care reprezintă numerele de ordine ale copiilor în ordinea în care au fost eliminați.

Exemplu:		ordine.in	ordine.out
6		2 4 1 3 5 6	

16. (Labirint - *)** Din nefericire un elf s-a pierdut în labirintul minotaurilor. Labirintul poate fi privit ca un caroiaj ale cărui celule pot fi libere sau ocupate de ziduri. Elful se poate deplasa pe verticală sau pe orizontală doar în celulele libere. Spre bucuria lui, a descoperit că nu a pierdut harta labirintului pe care o primise de la minotauri.

Fișierul de intrare *labirint.in* conține, pe prima linie, dimensiunile $m, n \leq 200$ ale labirintului. Următoarele m linii vor descrie celulele labirintului. Pe fiecare astfel de linie se vor afla câte n caractere. O celulă liberă va fi identificată prin caracterul "L", iar una ocupată prin caracterul "O". Ultima linie a fișierului va conține poziția inițială a elfului.

Fișierul de ieșire *labirint.out* trebuie să conțină, pe prima linie, lungimea minimă a unui drum care îl va duce pe elf la marginea labirintului. Cea de-a doua linie va conține o descriere a drumului. Fiecare mișcare a elfului va fi codificată printr-o literă. O mișcare de pe a i -a linie a labirintului pe cea de-a $(i-1)$ -a linie a acestuia va fi codificată prin "N". O mișcare de pe a i -a linie a labirintului pe cea de-a $(i+1)$ -a linie a acestuia va fi codificată prin "S". O mișcare de pe a i -a coloană a labirintului pe cea de-a $(i-1)$ -a coloană a acestuia va fi codificată prin "V". O mișcare de pe a i -a coloană a labirintului pe cea de-a $(i+1)$ -a coloană a acestuia va fi codificată prin "E". Se consideră că elful a reușit să iasă din labirint dacă reușește să ajungă pe prima linie, pe prima coloană, pe ultima linie sau pe ultima coloană. Se garantează faptul că va exista întotdeauna cel puțin o soluție.

Exemplu:

<i>labirint.in</i>		<i>labirint.out</i>
4 5		4
OLLOL		SVVS
OLOLO		
OLLO		
OLOOO		
2 4		

17. (Cifre - ***)** Eugenia îi pune adesea întrebări dificile lui Zăhărel, nu pentru că nu știe răspunsul, dar vrea să vadă cât de perspicace este Zăhărel. Uneori exagerează și întrebările ei sunt foarte grele, chiar și pentru Zăhărel; atunci acesta vă cere ajutorul vostru! Astăzi Eugenia i-a pus următoarea întrebare lui Zăhărel: "eu mă gândesc la un număr întreg din intervalul $[A...B]$ ($0 \leq A \leq B < 1.000.000.000$), care este probabilitatea ca numărul la care mă gândesc să conțină cel puțin K cifre de valoare C ?" Ajutați-l pe Zăhărel să răspundă cât mai repede la întrebare.

Pe prima linie din fișierul de intrare *cifre.in* se găsesc numerele întregi A, B, C și K (în ordinea aceasta)

Pe prima linie din fișierul de ieșire *cifre.out* se va scrie probabilitatea, un număr real cu patru zecimale, ca numărul din intervalul $[A...B]$ la care se gândește Eugenia să aibă cel puțin K cifre de valoare C .

Exemplu:

<i>cifre.in</i>		<i>cifre.out</i>
1 13 1 1		0.3846

(<http://infoarena.devnet.ro>)

18. (Triang - *)** Andreea a învățat la școală ce este un triunghi echilateral. Fascinată de aceste figuri geometrice, ea desenează în plan $N \leq 1.500$ puncte cu coordonate numere reale. Ea însă nu își dă seama câte triunghiuri echilaterale a desenat, așa că vă cere ajutorul vostru!

Pe prima linie a fișierului *triang.in* se află N . Pe următoarele N linii se vor afla coordonatele celor N puncte sub forma $x y$ ($-10.000 \leq x, y \leq 10.000$)

Pe prima linie a fișierului *triang.out* se va scrie numărul de triunghiuri echilaterale desenate de Andreea. Nu vor exista două puncte cu coordonate identice, iar orice punct poate fi folosit pentru formarea mai multor triunghiuri echilaterale. Pentru testarea egalității a două numere reale se recomandă folosirea unei precizii de 10^{-3}

Exemplu:

<i>triang.in</i>		<i>triang.out</i>
3		1
0 0		
4 0		
2 3.4641016		

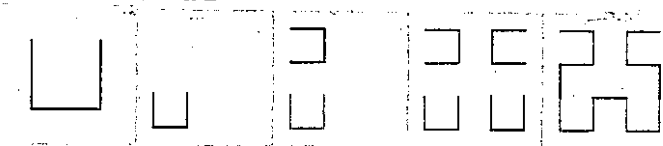
(<http://infoarena.devnet.ro>)

19. (Fractal - **) Hilbert a găsit o curbă care poate trece prin fiecare punct al spațiului, această curbă se bazează pe o construcție recursivă. Numim curbă de tip Hilbert de ordinul K , curba realizată după următoarele reguli, ce trece prin fiecare nod al unei grile de $2^K \times 2^K$ noduri și trece prin noduri vecine ale grilei. Curba Hilbert de ordinul 1 este o curbă simplă:

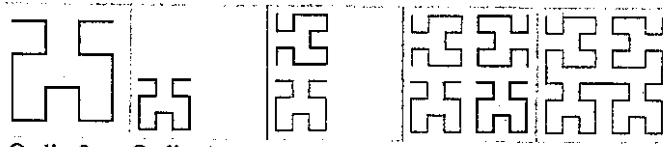


Vor fi descrise, în următoarele imagini, trecerile de la o curbă de ordin x la o curbă de ordin $x+1$:

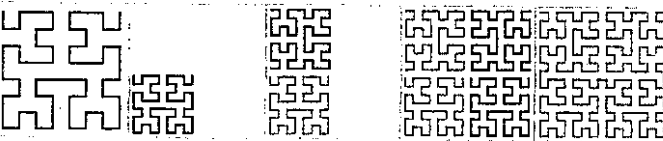
Ordin 1 -> Ordin 2



Ordin 2 -> Ordin 3



Ordin 3 -> Ordin 4



Ordin 4 -> Ordin 5



Se dau ca date de intrare din fișierul *fractal.in* numerele $K \leq 15$, x și y ($1 \leq x, y \leq 2^K$), unde K este ordinul unei curbe, iar x și y sunt coordonate întregi în interiorul unui pătrat de dimensiune $2^K \times 2^K$. Se cere să scrieți în fișierul de ieșire *fractal.out* în câți pași se ajunge la coordonatele (x, y) , dacă punctele din pătrat sunt parcurse în ordinea dată de curba Hilbert de ordin K . Coltul din stânga sus are coordonatele $(1, 1)$.

Exemplu:

<i>fractal.in</i>	<i>fractal.out</i>
3 2 3	13
2 4 1	15

(<http://infoarena.devnet.ro>)

20. (*Frac - *****) Pătrățul este mare pasionat de fracții. Într-o zi el se gândește să scrie pe o foaie de hârtie, în ordine crescătoare, toate fracțiile ireductibile cu numitorul N . Observând însă la timp că sunt o infinitate de astfel de fracții, el nu se mai obosește și dorește să afle doar a P -a fracție din șirul la care s-a gândit. Să se determine numărătorul celei de a P -a fracții din șirul construit după regulile de mai sus. Prima linie a fișierului *frac.in* conține două numere întregi $N \leq 12.000.000.000$ și $P \leq 10^{14}$, separate prin câte un spațiu, având semnificația descrisă în enunț. Prima linie a fișierului *frac.out* conține un număr natural care reprezintă numărătorul celei de a P -a fracții din șirul fracțiilor ireductibile cu numitorul N . Se garantează că rezultatul nu depășește 2^{61} .

Exemplu:

<i>frac.in</i>	<i>frac.out</i>
12 5	13

(<http://infoarena.devnet.ro>)

21. (*Compania - ****) Într-o companie cu N oameni, fiecare persoană este fie șef, fie angajat. Fiecare șef interacționează cu fiecare angajat din departamentul său (dar nu cu alți șefi). Gradul de eficiență al unui departament este egal cu numărul de interacțiuni distincte de tip șef - angajat, care au loc. Eficiența întregii companii este suma gradelor de eficiență al fiecărui departament. Se știe că eficiența companiei este $E \leq 2.000$, trebuie să se determine numărul minim de angajați, dintre care să fie un număr minim de șefi, pentru a se obține această eficiență. De asemenea, trebuie să se determine structura companiei, specificând numărul de departamente, și pentru fiecare departament numărul de angajați, și câți dintre aceștia sunt șefi.

Prima linie a fișierului *comp.in* conține numărul întreg E .

Prima linie a fișierului *comp.out* va conține numărul minim de oameni din companie, numărul minim de șefi și numărul de departamente din companie. Pe următoarele linii se vor găsi câte două numere naturale reprezentând numărul de oameni din acel departament și câți dintre aceștia sunt șefi.

Exemplu: *comp.in*

comp.out

7	7 3 2
	2 1
	5 2

(<http://infoarena.devnet.ro>)

22. (*Soldati - ****) Un copil are $N \leq 100$ soldați de jucărie, fiecare având ca înălțime un număr cuprins între 1 și N , iar soldații au înălțimi distincte. Zi de zi, copilul aranjează cei N soldați în linie, după placul său. De curând, are o nouă obsesie, și anume să aranjeze cei N soldați astfel încât să existe K secvențe monotone în aranjament. O secvență monotona este un șir de soldați aflați pe poziții consecutive în aranjament, iar înălțimea fiecărui soldat, mai puțin a primului din secvență, este mai mare decât înălțimea soldatului aflat în stânga lui, secvența fiind cea maximală cu această proprietate (adică nu poate fi adăugat în secvență un soldat astfel încât să se păstreze ordinea crescătoare a înălțimilor). Spre exemplu, pentru $N=5$ aranjamentul $(3 5 1 2 4)$ este format din două secvențe monotone $(3 5)$ și $(1 2 4)$. Determinați pentru N și K dat câte aranjamente posibile poate face copilul. Pe prima linie a fișierului text *soldati.in* se află numerele naturale N și K . Pe fiecare linie a fișierului de ieșire *soldati.out* se va scrie un singur număr reprezentând numărul total de aranjamente posibile.

Exemplu: *soldati.in*

soldati.out

3 2	4
-----	---

23. (*Lex - ****) Să considerăm n obiecte, numerotate de la 1 la $n \leq 30$. Din cele n obiecte se pot forma $2^n - 1$ submulțimi distincte nevide. Dacă ordonăm submulțimile lexicografic, putem asocia fiecărei submulțimi un număr de ordine de la 1 la $2^n - 1$. De exemplu, pentru $n=3$ submulțimile nevide în ordine lexicografică sunt:

$\{1\}, \{1, 2\}, \{1, 2, 3\}, \{1, 3\}, \{2\}, \{2, 3\}, \{3\}$

În multe aplicații practice este necesară implementarea eficientă a următoarelor două operații: dată fiind o submulțime, să se determine numărul ei de ordine; dat fiind un număr de ordine, să determine submulțimea corespunzătoare. Scrieți un program care să implementeze cele două operații.

Fișierul de intrare *lex.in* conține: pe prima linie numărul natural n ; pe a doua linie un număr natural $m \leq 1.000$, reprezentând numărul de operații ce trebuie executate; pentru fiecare operație urmează în fișier câte două linii; pe prima linie dintre cele două este scris tipul operației (1 sau 2); dacă tipul operației este 1, atunci pe cea de a doua linie este scris numărul de elemente din submulțime, urmat de elementele submulțimii, separate prin spații; dacă tipul operației este 2, pe cea de a doua linie va fi scris un număr de ordine (adică un număr natural cuprins între 1 și $2^n - 1$).

Fișierul de ieșire *lex.out* va conține m linii, câte una pentru fiecare operație din fișierul de intrare. Pe fiecare linie va fi scris rezultatul unei operații, în ordinea în care operațiile apar în fișierul de intrare. Dacă operația este de tip 1, în fișierul de ieșire va fi afișat numărul de ordine al submulțimii specificate în operație. Dacă operația este de tip 2, în fișierul de ieșire vor fi afișate elementele submulțimii cu numărul de ordine specificat, separate prin spații, în ordine crescătoare.

<i>Exemplu:</i>	<i>lex.in</i>		<i>lex.out</i>
3		1 2 3	
3		6	
2		3	
3			
1			
2 2 3			
2			
7			(http://campion.edu.ro)

24. (Windows - **) Vasile folosește un sistem de operare care deschide pe ecran numeroase ferestre. Ecranul este împărțit în pătrate elementare (care au aria 1x1) formând un caroiaj în care liniile sunt numerotate de la 1 de sus în jos, iar coloanele sunt numerotate de la 1 de la stânga la dreapta. Astfel, fiecare pătrat elementar de pe ecran poate fi identificat specificând numărul liniei și numărul coloanei pe care se află. Fiecare fereastră este un dreptunghi format din unul sau mai multe pătrate elementare. O fereastră nou deschisă poate să se suprapună (parțial sau total) peste alte ferestre, deschise în prealabil. Putem închide o fereastră dacă executăm un clic în pătratul elementar ce constituie colțul din dreapta sus al ferestrei (dacă acesta este vizibil). Scrieți un program care să determine numărul minim de click-uri necesare pentru a închide prima fereastră pe care am deschis-o. Fișierul de intrare *windows.in* conține, pe prima linie, un număr natural $N \leq 100$, reprezentând numărul de ferestre deschise pe ecran. Fiecare dintre următoarele N linii conține 4 numere naturale separate prin câte un spațiu $R1\ S1\ R2\ S2 \leq 10.000$, cu semnificația "am deschis o fereastră care are colțul din stânga sus pe linia $R1$ și coloana $S1$ și colțul din dreapta jos pe linia $R2$ și coloana $S2$ ". Ferestrele se deschid în ordinea în care apar în fișierul de intrare. Fișierul de ieșire *windows.out* conține o singură linie, pe care se află numărul minim de click-uri necesare pentru a închide prima fereastră deschisă.

<i>Exemplu:</i>	<i>windows.in</i>		<i>windows.out</i>
3		3	
4 1 6 3			
2 2 5 5			
1 4 3 6			

25. (Fotbal - *)** La selecția "Vreați să fiu mare!" participă $N \leq 1.000$ tineri fotbaliști. În urma testelor efectuate, pentru fiecare jucător s-au stabilit trei calificative reprezentând valoarea acestuia pe posturile de fundaș, mijlocas și atacant. Antrenorul dorește să selecteze 10 jucători de câmp din cei N (portarul va fi ales separat). Adept al sistemului de joc 4-4-2, antrenorul are nevoie de 4 fundași, 4 mijlocași și 2 atacanți. Evident, se dorește realizarea unei echipe competitive. Valoarea unei echipe constă în suma calificativelor jucătorilor, pe posturile pe care aceștia evoluează. Se pune problema selectării unei echipe cât mai performante.

Pe prima linie a fișierului *fotbal.in* se află N , reprezentând numărul de jucători. Pe următoarele N linii se află câte trei valori întregi din intervalul $[1, 10]$, separate prin spațiu, reprezentând calificativele jucătorilor pe posturile de fundaș, mijlocas și

atacant. În fișierul *fotbal.out* se va afișa valoarea celei mai competitive echipe ce se poate forma folosind jucătorii dați.

<i>Exemplu:</i>	<i>fotbal.in</i>		<i>fotbal.out</i>
15		91	
1 8 3			
7 7 9			
3 5 7			
10 9 8			
3 5 9			
4 6 2			
3 9 8			
6 4 4			
7 2 3			
10 4 9			
5 10 6			
9 8 2			
4 5 6			
8 2 1			
4 9 3			

26. (Coliniaritate - *)** Se consideră $N \leq 1.000$ puncte în plan. Să se determine mulțimea de cardinal maxim cu proprietatea că toate punctele pe care aceasta le conține sunt coliniare.

Numărul N , reprezentând numărul de puncte, se află pe prima linie a fișierului *colin.in*. Pe următoarele N linii se află câte două numere întregi, reprezentând coordonatele punctelor, valori din intervalul $[-10000, 10000]$.

Pe prima linie a fișierului *colin.out* se va afișa cardinalul mulțimii obținute. În continuare se vor afișa, câte unul pe linie, numerele de ordine ale punctelor conținute în această mulțime, sortate crescător.

<i>Exemplu:</i>	<i>colin.in</i>		<i>colin.out</i>
7		4	
10 20		2	
0 0		3	
44 90		5	
20 30		7	
22 45			
11 20			
66 135			

27. (Beculețe - *)** Pe un panou dreptunghiular sunt dispuse $N \cdot M$ beculițe. Asociat panoului cu becuri, există un panou de comutatoare pentru aprinderea/stingerea becurilor. Totuși, sistemul de funcționare al comutatoarelor este puțin bizar. Astfel, prin acționarea unui comutator se va schimba nu numai starea becului asociat acestuia, ci și starea becurilor vecine pe linie și coloană. Pe panou există becuri aprinse sau stinse. Pornind de la o configurație dată, se cere să se determine numărul minim de acționari de comutatoare care va realiza stingerea tuturor becurilor.

Pe prima linie a fișierului *beculete.in* se află N și M ($1 < N, M \leq 16$). Pe următoarele N linii ale fișierului se află, neseparate prin spații, câte M valori din mulțimea $\{0, 1\}$ reprezentând starea becurilor: 1-aprins, 0-stins.

În fișierul *beculete.out* se va afișa numărul minim de acționari de comutatoare. Dacă nu există soluție, în fișier se va afișa mesajul "NU EXISTA".

Exemplu: *beculete.in* *beculete.out*

```

4 4      4
0111
1100
0101
1110

```

28. (Divizori - **) Vom considera un număr natural N . În șirul A vom așeza toți divizorii lui N . Se cere să se permute elementele șirului A , astfel încât, pentru oricare două elemente consecutive $A[i]$ și $A[i+1]$ să avem fie $A[i]=A[i+1]*p$, fie $A[i+1]=A[i]*p$, unde p este un număr prim oarecare. Valoarea p poate diferi de la o pereche de elemente la alta.

Pe prima linie a fișierului *divizori.in* se află N ($2 \leq N \leq 2.000.000.000$).

Pe prima linie a fișierului *divizori.out* se va afișa lungimea șirului A . Pe a doua linie a fișierului se vor afișa elementele lui A . În cazul existenței mai multor soluții, se poate afișa oricare dintre ele.

Exemplu: *divizori.in* *divizori.out*

```

12      6
      1 2 4 12 6 3

```

29. (Vecini - ****) În Drumul Taberei există un bloc foarte ciudat. În primul an când a fost construit (să presupunem anul 1) avea un singur etaj, după care, în fiecare an, se construiește un nou etaj, astfel încât în anul X va avea X etaje. Dar acesta nu este singurul lucru ciudat. Modul în care este ocupat fiecare etaj este foarte ciudat. La etajul 1 (primul etaj începând numărătoarea de jos) stă tot timpul administratorul, deci este ocupat. De asemenea ultimul etaj, fiind nou, este tot timpul ocupat. Restul etajelor, însă, sunt ocupate (sau libere) după regulile:

- dacă anul trecut, etajul curent și etajul de dedesubt au fost ocupate, atunci etajul va fi liber anul acesta;

- dacă anul trecut, etajul curent și etajul de dedesubt au fost libere, atunci etajul va fi liber și anul acesta;

- dacă anul trecut, etajul curent a fost ocupat, iar cel de dedesubt a fost liber, atunci etajul curent va fi ocupat;

- dacă anul trecut, etajul curent a fost liber, iar cel de dedesubt a fost ocupat, atunci etajul curent va fi ocupat.

Scrieți un program care determină configurația etajelor după $N \leq 100.000$ ani de la construcție. Fișierul de intrare *vecini.in* conține un singur număr natural N , care reprezintă numărul de ani. În fișierul *vecini.out* se vor scrie N numere separate între ele printr-un spațiu. Numerele reprezintă starea fiecărui etaj, începând cu etajul 1. Reprezentarea este 0 pentru liber și 1 pentru ocupat.

Exemplu *vecini.in* *vecini.out*

```

5      1 0 0 0 1

```

("Stelele informaticii" 2003, București)

Indicații și răspunsuri

1.1. Șir de caractere

Sectiunea 1.1.1

1. c) d)	8. d)	15. b)	22. d)
2. a) b)	9. d)	16. b)	23. c)
3. a)	10. b)	17. c)	24. a)
4. b)	11. d)	18. c)	25. c)
5. a) c)	12. a) b) e)	19. c)	26. c)
6. c)	13. a)	20. a)	27. b)
7. d)	14. a)	21. b)	28. c)

Sectiunea 1.1.3

2.

```

1 var s:string;
2   x:char;
3 begin
4   readln(s); x:=s[1];
5   while pos(x,s)>0 do
6     delete(s,pos(x,s),1);
7   writeln(s);
8 end.

```

```

#include <string.h>
#include <iostream.h>
char s[256],x; int p;
void main() {
  cin>>s; x=s[0];
  while ((p=strchr(s,x)-s)>=0)
    strcpy(s+p,s+p+1);
  cout<<s<<endl;
}

```

3.

```

1 var s:string;
2 begin
3   readln(s);
4   while length(s)>1 do begin
5     delete(s,1,1);
6     delete(s,length(s),1);
7     writeln(s);
8   end;
9 end.

```

```

#include <string.h>
#include <iostream.h>
char s[256];
void main() {
  cin>>s;
  while (strlen(s)>1) {
    strcpy(s,s+1);
    s[strlen(s)-1]=0;
    cout<<s<<endl;
  }
}

```

7.

```

1 var x,m1,m2:string;
2   n,i:integer;
3 begin
4   readln(n);readln(m1);readln(m2);
5   if m2<m1 then begin
6     x:=m2;m2:=m1;m1:=x;
7   end;
8   for i:=3 to n do begin
9     readln(x);

```

```

#include <iostream.h>
#include <string.h>
char x[256],m1[256],m2[256];
int n,i;
void main() {
  cin>>n;cin>>m1;cin>>m2;
  if (strcmp(m1,m2)>0) {
    strcpy(x,m2);strcpy(m2,m1);
    strcpy(m1,x);
  }
}

```

```

10 if x<m1 then begin
11   m2:=m1;m1:=x; end
12 else if x<m2 then
13   m2:=x;
14 end;
15 writeln(m1+m2);
16 end.

```

```

10.
1 var s,x,y,s1,s2:string;
2   n,i:integer;
3 begin
4   readln(x); readln(y);
5   i:=length(x)+1; s1:='';
6   s:='aeiouAEIOU';
7   repeat
8     dec(i);
9     s1:=x[i]+s1;
10    until (i=1)or(pos(x[i],s)>0);
11    i:=length(y)+1;s2:='';
12    repeat
13      dec(i);
14      s2:=y[i]+s2;
15    until (i=1)or(pos(y[i],s)>0);
16    writeln(s1+s2);
17  end.

```

```

11.
1 var s:string;
2   i,j,er,cifra:integer;
3 begin
4   readln(s);
5   for i:=1 to length(s) do
6     begin
7       if s[i] in ['0'..'9'] then
8         val(s[i],cifra,er)
9       else
10        for j:=1 to cifra do
11          write(s[i]);
12    end;
13  end.

```

```

14.
1 var s,c,s1,s2:string; x:byte;
2 begin
3   readln(s);c:='';
4   readln(s1);readln(s2);
5   while(pos(s1,s)>0)do begin
6     x:=pos(s1,s);
7     c:=c+copy(s,1,x-1)+s2;
8     delete(s,1,x+length(s1)-1);
9   end;
10  writeln(c+s);
11 end.

```

```

for (i=3;i<=n;i++) {
  cin>>x;
  if (strcmp(x,m1)<0) {
    strcpy(m2,m1); strcpy(m1,x);}
  else if (strcmp(x,m2)<0)
    strcpy(m2,x);}
cout<<strcat(m1,m2)<<endl;}

```

```

#include <iostream.h>
#include <string.h>
char s[256],x[256],y[256],
s1[256],s2[256]; int i,n;
void main() {
  cin>>x;cin>>y; i=strlen(x);
  strcpy(s,"aeiouAEIOU");
  do { if (strlen(s1))
    memmove(s1+1,s1,strlen(s1)-1);
    s1[0]=x[--i];
  } while(i>0&&!strchr(s,x[i]));
  i=strlen(y);
  do { if (strlen(s2))
    memmove(s2+1,s2,strlen(s2)-1);
    s2[0]=y[--i];
  } while(i>0&&!strchr(s,y[i]));
  cout<<!strcmp(s1,s2)<<endl; }

```

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>
char s[256]; int i,j,cifra;
void main() {
  gets(s);
  for (i=0;i<strlen(s);i++)
    if (isdigit(s[i]))
      cifra=s[i]-'0';
  else
    for (j=0;j<cifra;j++)
      printf("%c",s[i]);
}

```

```

#include <stdio.h>
#include <string.h>
char s[256],c[256],s1[256],
s2[256]; int p;
void main() {
  gets(s);gets(s1);gets(s2);
  while((p=strstr(s,s1)-s)>=0){
    strncat(c,s,p);
    strcpy(s+p,s+p+strlen(s1));
    strcpy(s,s+p);strcat(c,s2);
    puts(strcat(c,s)); }

```

```

16.
1 var s:string;
2   i:byte; ok:boolean;
3 begin
4   readln(s);
5   for i:=1 to length(s)do
6     if ((upcase(s[i])>='A')and
7       (upcase(s[i])<='Z'))then
8       begin
9         write(s[i]); ok:=true;
10      end
11    else
12      if ok then begin
13        writeln(ok:=false);
14      end;
15  end.

```

```

20.
1 var s,cuv,t:string;
2   i:byte;
3 begin
4   readln(s);t:=''; cuv:='';
5   for i:=1 to length(s)do
6     if ((upcase(s[i])>='A')and
7       (upcase(s[i])<='Z'))then
8       cuv:=s[i]+cuv
9     else
10      begin
11        t:=t+cuv+s[i];
12        cuv:='';
13      end;
14    writeln(t);
15  end.
16

```

```

22.
1 var s:string; x:real;
2   er,i:integer;
3 begin
4   readln(s); val(s,x,er);
5   x:=length(s)-3;
6   if (er=0)and(pos('.',s)=x)
7     then write('Da')
8     else write('Nu');
9  end.

```

```

28.
1 var s,c,t:string;f,g:text;
2   sm,i,x,er:integer;
3 begin
4   assign(f,'virus.txt');reset(f);
5   assign(g,'sum.txt');rewrite(g);
6   c:='';

```

```

#include <stdio.h>
#include <string.h>
char s[256]; int i,ok;
void main() {
  gets(s);
  for (i=0;s[i];i++)
    if ((s[i]>='a'&&s[i]<='z')||
      (s[i]>='A'&&s[i]<='Z'))
      { putchar(s[i]); ok=1; }
    else
      if (ok){
        putchar('\n'); ok=0;
      }
}

```

```

#include <stdio.h>
#include <string.h>
char s[256],cuv[256],t[256];
int i;
void main() { gets(s);
  for (i=0;i<=strlen(s);i++)
    if ((s[i]>='a'&&s[i]<='z')||
      (s[i]>='A'&&s[i]<='Z')){
      if(strlen(cuv))
        memmove(cuv+1,cuv,strlen(cuv));
      cuv[0]=s[i]; }
    else {
      strcat(t,cuv);
      t[strlen(t)]=s[i];
      memset(cuv,0,sizeof(cuv)); }
  puts(t);
}

```

```

#include <string.h>
#include <stdio.h>
char s[256]; long x,y;
void main() {
  gets(s);
  if(sscanf(s,"%d.%d",&x,&y)==2&&
    (y>=100&&y<1000)) printf("Da\n");
  else printf("Nu\n");
}

```

```

#include <stdio.h>
#include <string.h>
char s[256]; FILE *f,*g;
int sm,i,c;
void main() {
  f=fopen("virus.txt","r");

```

```

7 while not eof(f) do begin
8   readln(f,s); sm:=0;
9   for i:=1 to length(s) do
10    if (s[i]>='0') and (s[i]<='9')
11     then c:=c+s[i]
12    else begin
13     val(c,x,er);
14     sm:=sm+x; c:='';
15     end;
16    val(c,x,er); c:='';
17    writeln(g,sm+x);
18  end;
19  close(g); close(f);
20 end.

```

```

g=fopen("sum.txt","w");
while (fscanf(f,"%s\n",s)==1){
  sm:=0;
  for(i:=0;i<=strlen(s);i++)
    if(s[i]>='0'&&s[i]<='9')
      c:=c*10+s[i]-'0';
    else { sm:=c; c:=0; }
  fprintf(g,"%d\n",sm);
}
fclose(f);fclose(g);
}

```

```

29.
1 var s,c,t:string;
2   i,p:integer;
3 begin
4   readln(s); readln(p); c:='';
5   for i:=1 to length(s) do
6     if (upcase(s[i])>='A') and
7       (upcase(s[i])<='Z')
8     then c:=c+s[i]
9     else begin
10      if length(c)<>p then
11        t:=t+c;
12      t:=t+s[i]; c:='';
13      end;
14      writeln(t);
15    end.
16

```

```

#include <string.h>
#include <stdio.h>
char s[256],c[256],t[256];
int i,p;
void main() {
  gets(s); scanf("%d",&p);
  for (i:=0;i<=strlen(s);i++)
    if ((s[i]>='A'&&s[i]<='Z') ||
        (s[i]>='a'&&s[i]<='z'))
      c[strlen(c)]=s[i];
    else {
      if (strlen(c)!=p) strcat(t,c);
      t[strlen(t)]=s[i];
      memset(c,0,sizeof(c));
      puts(t);
    }
}

```

```

30.
1 var s,c,y:string;
2   i,n,p:integer;
3 begin
4   readln(s); readln(p);
5   n:=length(s); c:='';
6   for i:=1 to n-p+1 do begin
7     y:=s; delete(y,i,p);
8     if pos(y,c)=0 then
9       c:=c+y+' ';
10    end;
11    writeln(c);
12  end.

```

```

#include <iostream.h>
#include <string.h>
char s[256],c[256],y[256];int i,n,p;
void main() {
  cin>>s>>p; n:=strlen(s);
  for(i:=0;i<=n-p;i++) {
    memset(y,0,sizeof(y));
    strncpy(y,s,i);
    strcat(y,s+i,p,n-i-p);
    if (!strstr(c,y))
      strcat(c, strcat(y, " ")); }
  cout<<c<<endl; }

```

```

33.
1 var s,c,y:string;
2   i,n,p,m:integer;
3 begin
4   readln(s); n:=length(s);
5   p:=1;
6   while p*(p+1)<=2*n do inc(p);
7   dec(p); m:=n-p*(p+1)div 2;

```

```

#include <iostream.h>
#include <string.h>
char s[256],c[256]; int i,n,p,m;
void main() {
  cin>>s; n:=strlen(s);
  for(p:=1;p*(p+1)<=2*n;p++); p--;
  m:=n-p*(p+1)/2;

```

```

8   writeln(copy(s,1,m));
9   delete(s,1,m);
10  while s<>'' do begin
11    writeln(copy(s,1,p));
12    delete(s,1,p);
13    dec(p);
14  end;
15 end.

```

```

cout<<strncat(c,s,m)<<endl;
strcpy(s,s+m);
while (strlen(s)) {
  memset(c,0,sizeof(c));
  cout<<strncat(c,s,p)<<endl;
  strcpy(s,s+p); p--;
}
}

```

```

34.
1 var a,b,c:string;
2   i,n,p,m:integer;
3 begin
4   readln(n); a:='a'; b:='b';
5   for i:=3 to n do begin
6     if odd(i) then c:=b+a
7     else c:=a+b;
8     a:=b;
9     b:=c;
10    end;
11    writeln(c);
12  end.
13

```

```

#include <iostream.h>
#include <string.h>
char a[256]={"a"},b[256]={"b"},
c[256]; int n,i;
void main() {
  cin>>n;
  for (i:=3;i<=n;i++){
    if (i%2==1){strcat(c,b);strcat(c,a);}
    else {strcat(c,a);strcat(c,b);}
    strcpy(a,b); strcpy(b,c);
    memset(c,0,sizeof(c));}
  cout<<b<<endl;
}

```

```

35.
1 var a:array[1..10] of string;
2   t:string;
3   i,n,j,max:integer;
4 begin
5   readln(n); max:=0;
6   for i:=1 to n do begin
7     readln(a[i]);
8     if length(a[i])>max then
9       max:=length(a[i]);
10    end;
11    for i:=1 to n-1 do
12      for j:=i+1 to n do
13        if a[i]>a[j] then begin
14          t:=a[i];a[i]:=a[j];a[j]:=t;
15          end;
16        for i:=1 to max do begin
17          for j:=1 to n do
18            if length(a[j])>=i then
19              write(a[j][i],' ');
20            else write(' ');
21          writeln; end; end.

```

```

#include <iostream.h>
#include <string.h>
char a[10][256],t[256];
int n,i,j,max;
void main() {
  cin>>n; max:=0;
  for (i:=0;i<n;i++) {
    cin>>a[i];
    if (max<strlen(a[i]))
      max:=strlen(a[i]);
  }
  for (i:=0;i<n;i++)
    for (j:=i+1;j<n;j++)
      if (strcmp(a[i],a[j])>0)
        {strcpy(t,a[i]);
         strcpy(a[i],a[j]);
         strcpy(a[j],t);}
  for (i:=0;i<max;i++){
    for (j:=0;j<n;j++)
      cout<<(i<strlen(a[j])?a[j][i]:' ')
      <<' '; cout<<endl; } }

```

```

38.
1 var i,j:byte;c,s:string;
2 begin
3   readln(s);
4   i:=2;
5   while i<length(s) do begin
6     if (s[i]+s[i+1]='at') and
7       (s[i-1]<>'.' ) and (s[i+2]<>'.' )

```

```

#include <iostream.h>
#include <string.h>
int i,j,p;
char c[256],s[256];
void main()
{
  cin>>s;

```

```

8 then begin
9   c:='';
10  for j:=1 to i-1 do c:=c+s[j];
11  c:=c+'e';
12  for j:=i+2 to length(s) do
13    c:=c+s[j];
14  writeln(c);
15  delete(c,pos('cod',c),3);
16  writeln(c);
17  end;
18  inc(i);
19  end;
20 end.

```

```

for (i=1;i<strlen(s);i++){
  if (s[i]=='a'&&s[i+1]=='t'&&
      s[i-1]!='.'&&s[i+2]!='.') {
    memset(c,0,sizeof(c));
    strcpy(c,s,i);strcat(c,"e");
    strcat(c,s,i+2,strlen(s)-i-2);
    cout<<c<<endl;
    p=strstr(c,"cod")-c;
    strcpy(c+p,c+p+3);
    cout<<c<<endl;
  }
}

```

```

22 for i:=1 to length(a[pr]) do
23   s:=s+ord(a[pr][i]);
24 for i:=1 to n do
25   if (a[i]='')or(i=pr) then
26     writeln(s+i)
27   else writeln(a[i])
28 end.

```

```

for (i=0;i<strlen(a[pr]);i++)
  s+=a[pr][i];
for (i=0;i<n;i++)
  if (!strlen(a[i])||i==pr)
    cout<<s+i+1<<endl;
  else cout<<a[i]<<endl;
}

```

1.2. Tipul înregistrare – structură

Sectiunea 1.2.1

1. a) c)	5. b) d)	9. a) c)	13. b)
2. d)	6. a) c)	10. d)	14. b)
3. a) d)	7. a) d)	11. c)	
4. c)	8. a) b)	12. c)	

Sectiunea 1.2.3

```

39.
1 var i,j:byte;c,s,p:string;
2 begin
3   readln(s); i:=1; p:='';
4   while i<length(s) do
5     if (s[i]='(') then begin
6       c:='';inc(i);
7       while s[i]<>')' do begin
8         c:=c+s[i]; inc(i);
9       end;
10      for j:=1 to ord(s[i+1])-48 do
11        p:=p+c; inc(i,2);
12      end
13      else begin
14        p:=p+s[i]; inc(i)
15      end;
16      writeln(p);
17    end.

```

```

#include <iostream.h>
#include <string.h>
int i,j;
char s[256],p[256],c[256];
void main() {
  cin>>s;
  for(i=0;i<strlen(s);i++){
    if(s[i]=='('){
      memset(c,0,sizeof(c));
      for (i++;s[i]!=')';i++)
        c[strlen(c)]=s[i];
      for (j=0;j<s[i+1]-'0';j++)
        strcat(p,c); i++;
    }
    else p[strlen(p)]=s[i];
    cout<<p<<endl;
  }
}

```

```

47.
1 var s,n,j,i,p,pr:integer;
2   x:string;ok:boolean;
3   a:array[1..10]of string;
4 begin
5   readln(n); i:=1; ok:=true;
6   while i<=n do begin
7     readln(x); p:=1;
8     for j:=1 to length(x) do
9       if x[j]>x[p] then p:=j;
10    delete(x,1,p-1);
11    for j:=1 to i-1 do
12      if a[j]=x then p:=0;
13    if p=0 then a[i]:='';
14    else begin
15      if ok then begin
16        pr:=i; ok:=false;
17      end;
18      a[i]:=x;
19    end;
20    inc(i)
21  end;

```

```

#include <iostream.h>
#include <string.h>
int s,n,i,j,p,pr,ok;
char x[256],a[11][256];
void main() {
  cin>>n;
  ok=1;
  for (i=0;i<n;i++){
    cin>>x;
    for (p=j=0;j<strlen(x);j++)
      if (x[j]>x[p]) p=j;
    strcpy(x,x+p);
    for (p=1,j=0;j<i;j++)
      if (!strcmp(a[j],x)) p=0;
    if (p==0)
      memset(a[i],0,sizeof(a[i]));
    else {
      if (ok) { pr=i; ok=0; }
      strcpy(a[i],x);
    }
  }
}

```

```

1.
1 type e=record
2   np:string; m:real;
3   end;
4 var a:array[1..50]of e;
5 x:e;
6 nt,j,n,i:integer;
7 mo,mx,s:real;
8 nm,pr:string;
9 begin
10  readln(n);
11  for i:=1 to n do begin
12    readln(nm);
13    readln(pr);
14    readln(mo,nt);
15    with a[i] do begin
16      np:=nm+' '+pr;
17      m:=(mo*3+nt)/4;
18      s:=s+m;
19      if m>mx then mx:=m;
20    end;
21  end;
22  writeln(s/n:0:2);
23  for i:=1 to n do
24    if a[i].m=mx then
25      writeln(a[i].np);
26 end.

```

```

#include <stdio.h>
#include <string.h>
typedef struct {
  char np[256];
  float m;
} e;
e a[50];
int n,i;
float mo,mx,nt,s;
char nm[256],pr[256];
void main() {
  scanf("%d",&n);
  for (i=0;i<n;i++) {
    scanf("%s%f",nm,pr,&mo,&nt);
    strcpy(a[i].np,nm);
    strcat(a[i].np," ");
    strcat(a[i].np,pr);
    a[i].m=(mo*3.0+nt)/4.0;s+=a[i].m;
    if (mx<a[i].m) mx=a[i].m;
  }
  printf("%.2f\n",s/n);
  for(i=0;i<n;i++)
    if (a[i].m==mx)
      printf("%s\n",a[i].np);
}

```


3.

```

1  type e=record
2      np:string; m:real;
3  end;
4  var a:array[1..50] of e;
5  x:e; nt,j,n,i:integer;
6  mo:real;
7  nm,pr:string;
8  begin
9      readln(n);
10     for i:=1 to n do begin
11         readln(nm); readln(pr);
12         a[i].np:=nm+' '+pr;
13         readln(mo,nt);
14         a[i].m:=(mo*3+nt)/4;
15     end;
16     for i:=1 to n-1 do
17         for j:=i+1 to n do
18             if a[i].np>a[j].np then
19                 begin
20                     x:=a[i];
21                     a[i]:=a[j]; a[j]:=x;
22                 end;
23         for i:=1 to n do
24             if a[i].m<4.5 then
25                 writeln(a[i].np);
26     end.

```

6.

```

1  type e=record
2      np:string; vm:real;
3  end;
4  var a:array[1..50] of e;
5  x:e; nt,j,n,i,l:integer;
6  mv,v,m:real; nm,pr:string;
7  begin
8      readln(n,l);
9      for i:=1 to n do
10         begin
11             readln(nm); readln(pr);
12             a[i].np:=nm+' '+pr;
13             readln(m,v);
14             a[i].vm:=v/m;
15         end;
16         for i:=1 to n-1 do
17             for j:=i+1 to n do
18                 if a[i].np>a[j].np then
19                     begin
20                         x:=a[i]; a[i]:=a[j];
21                         a[j]:=x;
22                     end;
23         for i:=1 to n do
24             if a[i].vm>L then
25                 writeln(a[i].np);
26     end.

```

```

#include <stdio.h>
#include <string.h>
typedef struct {
    char np[256]; float m;
} e;
e a[50]; x; int n,i,j;
float mo,nt; char
nm[256],pr[256];
void main() {
    scanf("%d",&n);
    for (i=0;i<n;i++) {
        scanf("%s%f",nm,pr,&mo,&nt);
        strcpy(a[i].np,nm);
        strcat(a[i].np," ");
        strcat(a[i].np,pr);
        a[i].m=(mo*3.0+nt)/4.0;
    }
    for (i=0;i<n;i++)
        for (j=i+1;j<n;j++)
            if(strcmp(a[i].np,a[j].np)>0){
                x=a[i]; a[i]=a[j]; a[j]=x;
            }
    for (i=0;i<n;i++)
        if (a[i].m<4.5)
            printf("%s\n",a[i].np);
}

```

```

#include <stdio.h>
#include <string.h>
typedef struct {
    char np[256];
    float vm;
} e;
e a[50]; x;
int n,i,j,l;
float v,m; char nm[256],pr[256];
void main() {
    scanf("%d%d",&n,&l);
    for (i=0;i<n;i++)
        (scanf("%s",nm,pr);
         strcpy(a[i].np,nm);
         strcat(a[i].np," ");
         strcat(a[i].np,pr);
         scanf("%f",&v,&m);
         a[i].vm=v/m; )
    for (i=0;i<n;i++)
        for (j=i+1;j<n;j++)
            if(strcmp(a[i].np,a[j].np)>0){
                x=a[i]; a[i]=a[j]; a[j]=x;
            }
    for (i=0;i<n;i++)
        if (a[i].vm>l)
            printf("%s\n",a[i].np);
}

```

7.

```

1  type e=record
2      nr,nm:integer;
3  end;
4  var a:array[1..50] of e;
5  j,n,i,nr:integer;
6  v:real;
7  begin
8      readln(n); nr:=0;
9      for i:=1 to n do begin
10         readln(a[i].nr);
11         readln(a[i].nm);
12     end;
13     v:=a[n].nr/a[n].nm;
14     for i:=1 to n-1 do
15         if a[i].nr/a[i].nm=v then
16             inc(nr);
17     writeln(nr);
18 end.

```

10.

```

1  type e=record x,y:integer end;
2  var p:array[1..50] of e;
3  j,n,i,nr,max,a:integer;
4  x:e;
5  begin
6      readln(n); max:=0;
7      for i:=1 to n do
8         read(p[i].x,p[i].y);
9         for i:=1 to n-1 do
10             for j:=i+1 to n do
11                 if (p[i].x>p[j].x) or
12                     (p[i].x=p[j].x) and
13                     (p[i].y>p[j].y)
14                 then begin
15                     x:=p[i];
16                     p[i]:=p[j]; p[j]:=x;
17                 end;
18         for i:=1 to n do begin
19             j:=i+1; nr:=1;
20             while (p[j].x-p[i].x<=n)
21                 and (j<=n) do begin
22                 if (p[j].y-p[i].y<=n) and
23                     (p[j].y-p[i].y>0) then
24                     inc(nr);
25                 inc(j);
26             end;
27             if nr > max then begin
28                 max:=nr;
29                 a:=i;
30             end;
31         end;
32     write(p[a].x,' ',p[a].y)
33 end.

```

```

#include <stdio.h>
typedef struct {
    int nr,nm;
} e;
e a[50];
int n,i,j,nr;
void main()
{
    scanf("%d",&n); nr:=0;
    for (i=0;i<n;i++)
        scanf("%d",&a[i].nr,&a[i].nm);
    for (i=0;i<n;i++)
        if (a[i].nr*a[n-1].nm==
            a[n-1].nr*a[i].nm)
            nr++;
    printf("%d\n",nr);
}

```

```

#include <stdio.h>
typedef struct {int x,y;} e;
e p[50]; x;
int n,i,j,nr,max,a;
void main()
{
    scanf("%d",&n); max:=0;
    for (i=0;i<n;i++)
        scanf("%d",&p[i].x,&p[i].y);
    for (i=0;i<n;i++)
        for (j=i+1;j<n;j++)
            if ((p[i].x>p[j].x) ||
                (p[i].x==p[j].x && p[i].y>p[j].y))
            {
                x=p[i];
                p[i]=p[j];
                p[j]=x;
            }
    for (i=0;i<n;i++)
        for (j=i+1;j<n;j++)
            if (p[j].x-p[i].x<=n &&
                p[j].y-p[i].y<=n &&
                p[j].y-p[i].y>0) nr++;
    if (nr>max) {
        max=nr;
        a=i;
    }
    printf("%d %d\n",p[a].x,p[a].y);
}

```

11.

```

1 type e=record x,y:integer end;
2 var p:array[1..50] of e;
3 j,n,i,nr,max,a:integer; x:e;
4 begin
5 readln(n);
6 for i:=1 to n do
7   read(p[i].x,p[i].y);
8 for i:=1 to n-1 do
9   for j:=i+1 to n do
10    if (p[i].y>p[j].y) then begin
11      x:=p[i];p[i]:=p[j];p[j]:=x
12    end;
13 max:=0; nr:=1;
14 for i:=2 to n do
15   if (p[i-1].y=p[i].y) then
16     inc(nr)
17   else begin
18     if nr>max then max:=nr;
19     nr:=1;
20   end;
21 if nr>max then max:=nr;
22 writeln(max)
23 end.

```

13.

```

1 type e=record
2   nr,nm:integer; v:real end;
3 var a:array[1..50] of e;
4 j,x,y,n,i,nr,m:integer;
5 t:e; f,g:text;
6 begin
7 assign(f,'in.txt'); reset(f);
8 assign(g,'out.txt'); rewrite(g);
9 readln(f,n); m:=0;
10 for i:=1 to n do begin
11   inc(m);
12   read(f,a[m].nr,a[m].nm);
13   a[m].v:=a[m].nr/a[m].nm;
14   with a[m] do begin
15     x:=nr; y:=nm;
16     while x<>y do
17       if x>y then x:=x-y
18       else y:=y-x;
19     if x=1 then dec(m);
20   end;
21 end;
22 for i:=1 to m-1 do
23   for j:=i+1 to m do
24     if (a[i].v>a[j].v) then begin
25       t:=a[i]; a[i]:=a[j]; a[j]:=t
26     end;
27 for i:=1 to m do
28   writeln(g,a[i].nr,'/',a[i].nm);
29 close(g); end.

```

```

#include <stdio.h>
typedef struct {int x,y;} e;
e p[50],x;
int n,i,j,nr,max,a;
void main() {
  scanf("%d",&n);
  for (i=0;i<n;i++)
    scanf("%d%d",&p[i].x,&p[i].y);
  for (i=0;i<n;i++)
    for (j=i+1;j<n;j++)
      if (p[i].y>p[j].y)
        { x=p[i]; p[i]=p[j]; p[j]=x; }
  max=0;
  nr=1;
  for (i=1;i<n;i++)
    if (p[i-1].y==p[i].y) nr++;
  else {
    if (nr>max) max=nr;
    nr=1; }
  if (nr>max) max=nr;
  printf("%d\n",max);
}

```

```

#include <stdio.h>
typedef struct {
  int nr,nm; float v;
} e;
e a[50],t; int n,i,j,x,y,nr,m;
FILE *f,*g;
void main() {
  f=fopen("in.txt","r");
  g=fopen("out.txt","w");
  fscanf(f,"%d",&n);m=0;
  for (i=0;i<n;i++)
  {
    fscanf(f,"%d%d",&a[i].nr,&a[i].nm);
    a[i].v=a[i].nr/(float)a[i].nm;
    x=a[i].nr;y=a[i].nm;
    while (x!=y)
      if (x>y) x-=y; else y-=x;
    if (x==1) m--; m++;
  }
  for (i=0;i<m;i++)
    for (j=i+1;j<m;j++)
      if (a[i].v>a[j].v) {
        t=a[i];a[i]=a[j];a[j]=t;
      }
  for (i=0;i<m;i++)
    fprintf(g,"%d/%d\n",a[i].nr,
    a[i].nm);
  fclose(g);
}

```

16.

```

1 type e=record
2   s:byte;x,y:real;
3 end;
4 var p:array[1..50] of e;
5 j,n,i,nr:integer;
6 v1,v2:real;
7 x:e;
8 f,g:text;
9 begin
10 assign(f,'film.txt'); reset(f);
11 readln(f,n);
12 v1:=0;
13 v2:=24;
14 for i:=1 to n do begin
15   readln(f,p[i].x,p[i].y);
16   p[i].s:=i;
17   if p[i].x>v1 then v1:=p[i].x;
18   if p[i].y<v2 then v2:=p[i].y;
19 end;
20 for i:=1 to n-1 do
21   for j:=i+1 to n do
22     if (p[i].x>p[j].x) then begin
23       x:=p[i];
24       p[i]:=p[j];
25       p[j]:=x
26     end;
27 writeln(v1:0:2,v2:6:2);
28 for i:=1 to n do
29   write(p[i].s,' ');
30 end.

```

```

#include <stdio.h>
typedef struct {
  int s; float x,y;
} e;
e p[50],x;
int n,i,j,nr;
float v1,v2,xx,yy;
FILE *f;
void main() {
  f=fopen("film.txt","r");
  fscanf(f,"%d",&n);v1=0;v2=24;
  for (i=0;i<n;i++){
    fscanf(f,"%f%f",&xx,&yy);
    p[i].x=xx;
    p[i].y=yy;
    p[i].s=i;
    if (p[i].x>v1) v1=p[i].x;
    if (p[i].y<v2) v2=p[i].y;
  }
  for (i=0;i<n;i++)
    for (j=i+1;j<n;j++)
      if (p[i].x>p[j].x) {
        x=p[i];
        p[i]=p[j];
        p[j]=x;
      }
  printf("%.2f %.2f\n",v1,v2);
  for (i=0;i<n;i++)
    printf("%d",p[i].s+1);
}

```

Secțiunea 1.3.2

1. Se împarte textul în cuvinte care se vor insera într-un vector, păstrând mereu numai cuvinte distincte. La final, se sortează vectorul după criteriul din enunț și se afișează cuvintele.

2. Se poate genera matricea fiș prin completarea succesivă a fiecărei diagonale, fie folosind "formula": $M[i, j] = |i - j|$.

3. Ca să calculăm puterea la care apare factorul prim p în descompunerea lui $n!$ se poate folosi următoarea formula: $f = [n/p] + [n/p^2] + [n/p^3] + \dots$ ($[]$ reprezintă partea întreagă). Având aceasta formulă, se poate traversa un anumit rând din triunghiul lui Pascal și pentru fiecare element (r, c) , se poate calcula puterea la care apare D în descompunerea lui $r! / ((r-c)! * c!)$. Atunci când D nu este prim, trebuie avut grijă să se verifice dacă respectivul element (r, c) are în descompunerea sa toți factorii primi ai lui D . Dacă $D=6$, elementul din (r, c) trebuie să conțină 2 și 3 în descompunerea sa, iar dacă $D=4$, trebuie să conțină 2 de cel puțin două ori.

O altă modalitate de a calcula puterea la care apare un factor prim p în descompunerea unui număr este: fie $A[c] =$ puterea la care apare p în descompunerea lui (r, c) , $A[c+1] = A[c] +$ puterea lui p în $(r-c) \cdot$ puterea lui p în $(c+1)$. Această relație se poate deduce din modul în care se calculează elementul $(r, c+1)$ din (r, c) folosind formula $(r, c) = r! / ((r-c)! \cdot c!)$.

4. Se parcurge fișierul caracter cu caracter (nu este necesară stocarea datelor de intrare într-un vector) și se mențin două variabile care indică poziția de început și sfârșit a ultimului cuvânt detectat până în prezent, dacă s-a găsit vreunul. De asemenea, se păstrează și două variabile pentru suma lungimilor cuvintelor și numărul de cuvinte pentru a calcula rezultatul. Atenție însă la sfârșitul parcurgerii fișierului de ieșire, dacă ultimul caracter citit a fost o literă mare sau mică, să se actualizeze numărul de cuvinte și suma lungimilor.

5. Se simulează algoritmi de criptare și decriptare descriși în enunț.

6. Se împarte textul în cuvinte și se inserează într-un vector de cuvinte; când cuvântul există deja în vector se scrie poziția din vector.

7. Problema se rezolvă în mod asemănător cu problema "Camion" de la secțiunea 1.3.1 (probleme rezolvate), deoarece sunt necesare doar rezultatele pentru primele 10 linii.

8. La fiecare pas se scade din n cel mai mare număr Fibonacci mai mic decât n . Se implementează numere mari deoarece n poate avea 80 de cifre.

9. Se parcurge matricea pe coloane și se determină câte blocuri apar pe fiecare coloană.

10. Dacă șirul are lungime 1 sau 3, rezolvarea este trivială. Pentru lungime mai mare decât 3 se verifică primul caracter din șir și se determină dacă începe cu B sau C . Se păstrează o stivă cu caracterele recunoscute până acum, și se parcurge șirul caracter cu caracter. O perla B va fi mereu de forma 22...221A3AC, iar o perla C va fi de forma 2, 12A sau 322...221A3ACC. Cum perla A poate fi orice caracter din șir, problema se reduce la împărțirea șirului în perle de tip C și secvențe de cifre în funcție de caracterul din vârful stivei.

11. Șirul se determină după următoarea regulă:

1 (unu)

11 (un, unu)

21 (doi, de unu)

1211 (un doi, un unu)

111221 (un unu, un doi, doi de unu)

312211 (trei de unu, doi de doi, un unu)

...

Se poate demonstra că șirul va fi mereu format din cifrele 1, 2 și 3.

12. Problema se rezolvă relativ simplu, ținând cont de următoarele observații:

1) Deoarece întregul mesaj a fost codificat prin 4 rotiri ale textului, este clar că la o poziționare a textului sub șablon pot fi citite $Lung(Mesaj)/4$ caractere, deci întregul mesaj are $4 \cdot NumarGăuri$ caractere.

2) Ca urmare a observației de la punctul 1, mesajul poate fi împărțit exact în 4 șiruri de lungimi egale $Mesaj1, \dots, Mesaj4$.

3) Dacă o gaură se află în poziția $T[i, j]$ din șablon, ei îi corespund pozițiile

$T[j, N-i+1]$ la rotire cu 90 grade

$T[N-i+1, N-j+1]$ la rotire cu 180 grade

$T[N-j+1, i]$ la rotire cu 270 grade

de unde deducem că nu e nevoie să rotim textul.

4) Dacă lungimea unui șir este $L4$, este suficient să parcurgem numai primul dintre cele 4 șiruri cu un *Index*. Atunci, parcurgând textul care ascunde mesajul, în poziția (i, j) există o gaură în șablon dacă și numai dacă toate cele 4 caractere $Mesaj1[Index], Mesaj2[Index], Mesaj3[Index], Mesaj4[Index]$ coincid cu cele 4 caractere obținute prin rotire (vezi observația 3).

13. Se înlocuiesc literele din cuvinte cu cifre de la 0 la 9, în funcție de poziția în alfabet. Cum pot fi doar 10.000 de cuvinte distincte, se va folosi un algoritm de sortare prin numărare.

14. Se împarte textul în cuvinte și se distribuie pe linii de lungime *Max* conform regulilor descrise în enunț.

15. Se va folosi algoritmul "clasic" de determinare a celui mai lung subșir comun a două șiruri folosind *programarea dinamică*. Ca să se determine o soluție cât mai mare din punct de vedere al valorii, se va menține pe lângă matricea $C[i, j] =$ lungimea celui mai lung subșir comun din șirurile $n[i...lungime(n)]$ și $m[j...lungime(m)]$ o matrice $V[i, j] =$ cea mai mare valoare pentru cel mai lung subșir comun din șirurile respective. Relațiile de recurență vor fi:

dacă $n[i] = m[j]$ atunci $C[i, j] \leftarrow C[i+1, j+1] + 1$
 $V[i, j] \leftarrow n[i] + V[i+1, j+1]$ (+ reprezintă concatenare)

altfel
dacă $C[i+1, j] > C[i, j+1]$ atunci $C[i, j] \leftarrow C[i+1, j]$
 $V[i, j] \leftarrow V[i+1, j]$

altfel
dacă $C[i+1, j] < C[i, j+1]$ atunci $C[i, j] \leftarrow C[i, j+1]$
 $V[i, j] \leftarrow V[i, j+1]$

altfel
dacă $(C[i+1, j] = C[i, j+1]) \text{ AND } (V[i+1, j] > V[i, j+1])$ atunci
 $C[i, j] \leftarrow C[i+1, j]$
 $V[i, j] \leftarrow V[i+1, j]$

altfel
dacă $(C[i+1, j] = C[i, j+1]) \text{ AND } (V[i+1, j] < V[i, j+1])$ atunci
 $C[i, j] \leftarrow C[i, j+1]$
 $V[i, j] \leftarrow V[i, j+1]$

16. Se va folosi algoritmul "clasic" de determinare a celui mai lung subșir comun a două șiruri și anume, între șirul normal și șirul inversat. Rezultatul va fi numărul de caractere din șir – lungimea celui mai lung subșir comun. Pentru exemplul din enunț: $CMLSC("Abxbd", "dbxbA") = 3$, astfel răspunsul va fi $5-3 = 2$ (se va insera un "A" și un "d" pentru a se obține "AdbxbdA").

17. Numim vecinii unei instrucțiuni i , instrucțiunile la care se poate ajunge după executarea instrucțiunii i . Din definiția programului, fiecare instrucțiune are maximum 2 vecini. Pentru a verifica ce instrucțiuni nu se execută, vom folosi o structură de date de tip *coadă*, numită astfel deoarece este asemănătoare cu o "coadă" (la magazin, la farmacie, etc. ☺). Coada va avea un început și un sfârșit, iar pe măsura ce se inserează elemente, se inserează la sfârșit; dacă se șterg elemente, se șterg de la început. Astfel, inițial se inserează instrucțiunea 1 în coadă și se repetă următorii pași:

- fie i elementul de la începutul cozii; se vor insera în coadă toți vecinii lui i care nu au fost deja inserați la un moment dat până acum;
- se șterge i din coadă;
- dacă coada este goală, atunci stop.

Astfel, folosind acest mecanism se vor determina toate instrucțiunile care se execută, cele care nu se execută fiind exact acelea care nu au fost inserate niciodată în coadă.

18. Cele 5 valori care trebuie calculate se vor reține într-un vector cu 5 elemente: $v[1]$ pentru numărul de caractere 'T', $v[2]$ pentru numărul de caractere 'V', $v[3]$ pentru numărul de caractere 'X', $v[4]$ pentru numărul de caractere 'L' și $v[5]$ pentru numărul de caractere 'C'.

Programul parcurge cele n valori și determină, pentru fiecare, numărul de caractere folosite prin determinarea intervalului căruia îi aparține numărul. Se disting cazurile: [100..399], [90..99], [50..89], [40..49], [10..39], [9], [5..8], [4], [1..3].

Pentru fiecare caz în parte, se incrementează contoarele necesare și se scade valoarea care reprezintă capătul inferior al intervalului. Calculul continuă până când valoarea respectivă devine 0.

19. Numim *coadă cu dublă prioritate* o structură de date care suportă următoarele operații:

- inserare element la sfârșitul cozii;
- inserare element la începutul cozii;
- extragere element de la sfârșitul cozii;
- extragere element de la începutul cozii;
- accesarea elementului de la începutul cozii;
- accesarea elementului de la sfârșitul cozii.

Se vor utiliza două cozi cu dublă prioritate:

- 1) Coada a cu elemente de tip caracter, în care se vor insera succesiv caracterele din șirul inițial și ulterior cele specificate în operațiile din fișierul de intrare.
- 2) Coada b cu elemente de tip întreg; $b[i]$ = lungimea secvenței regulate care începe cu poziția i . Se vor implementa cozile a și b ca doi vectori.

Pentru a putea realiza operații și la începutul și la sfârșitul vectorilor, se va dubla spațiul de memorie necesar pentru fiecare dintre cei doi vectori și se începe inserarea elementelor de la mijloc. Inițial, se inserează în ordine toate caracterele șirului citit în coada a . Pentru a păstra ordinea, fiecare nou caracter va fi inserat la sfârșitul cozii. Când se inserează o paranteză închisă la sfârșitul cozii, iar pe ultima poziție în coadă este paranteza deschisă corespunzătoare, se deduce că se formează o secvență regulată cu lungimea 2 mai mare decât cea deja existentă pe această poziție. Prin urmare, se reține în vectorul b această lungime și se elimină din coada a cele două paranteze. Când se inserează la începutul cozii a o paranteză deschisă, iar la începutul cozii a era paranteza închisă corespunzătoare, se formează o secvență regulată cu 2 mai mare și se execută aceleași operații ca în cazul precedent.

20. Se va memora într-un vector de șiruri de caractere numerele de înregistrare ale mașinilor în ordinea intrării lor în tunel. După citirea acestui vector, mașinile vor fi identificate prin poziția lor în vector (un număr cuprins între 0 și $N-1$, unde N este numărul de mașini). La citirea informațiilor despre ordinea de ieșire a mașinilor din tunel se va construi un vector denumit *pozOut* în care se va reține numerele de identificare ale mașinilor care ies din tunel. Pentru a determina numărul de șoferi care au efectuat depășiri în tunel, este suficient să se determine numărul de inversiuni din acest vector. Mai exact dacă $i < j$ și $pozOut[j] < pozOut[i]$, a avut loc o depășire.

21. Se concatenează primul șir cu el însuși și se caută prima apariție a celui de-al doilea șir în șirul nou format.

22. Se aduce numărul la o valoare inferioară lui 500000000, reținând în șirul în care se construiește reprezentarea super-romană 1, 2 sau 3 caractere 'Z'. Numărul rămas (< 500000000) este împărțit în cifre corespunzătoare ordinului de mărime (unitatea, zeci, sute,...). Toate cifrele se tratează la fel, dar în funcție de poziția lor în număr (unitatea, zeci, ...), rezultatul va fi diferit. Importanța este deci, în acest caz, organizarea simbolurilor folosite. Astfel, acestea au fost organizate în perechi corespunzătoare ordinului de mărime: I, V pentru unități; X, L pentru zeci; C, D pentru sute; M, P pentru mii; Q, R pentru zeci de mii; S, T pentru sute de mii; U, B pentru milioane; W, T pentru zeci de milioane; Y, Z pentru sute de milioane. La parcurgerea șirului de cifre vor fi tratate cele 9 cazuri, care vor da rezultate diferite în funcție de ordinul de mărime al cifrei respective.

23. Se observă că pentru a scrie numerele din intervalul $[10^{x-1}, 10^x - 1]$ sunt necesare $x \cdot 9 \cdot 10^{x-1}$ cifre. Se parcurg succesiv toate intervalele de forma $[10^{x-1}, 10^x - 1]$. Dacă $n > 10^x - 1$ se adună la numărul total de cifre necesare $x \cdot 9 \cdot 10^{x-1}$ (numărul de cifre necesar pentru a scrie toate cifrele din acest interval) și se trece la următorul interval. Dacă $n \leq 10^x - 1$, atunci din intervalul curent mai trebuie să se adune numărul de cifre necesare pentru scrierea numerelor din intervalul $[10^{x-1}, n]$, adică $x \cdot (n - 10^{x-1} + 1)$ cifre.

24. Fie S_i șirul rotit care începe la indexul i . Trebuie determinată rotația care este minimă în ordine lexicografică. Se vor compara șirurile într-un mod în care se organizează confruntările la un turneu de tenis, câștigătorul turneului fiind rotația care este minimă în ordine lexicografică. La un pas al turneului, se realizează următoarele operații:

Fie șirul inițial împărțit în trei secvențe ABC. La pasul actual, trebuie comparat șirul ABC cu șirul BCA, unde A și B au aceeași lungime:

Dacă $A < B$, atunci ABC este câștigătorul meciului dintre ABC și BCA.

Dacă $A > B$, atunci BCA este câștigătorul meciului dintre ABC și BCA.

Dacă $A = B$, atunci se ia ABC ca și câștigător; dacă decizia e greșită și $ABC > BCA$, atunci înseamnă că $C < A$ deci la un moment ulterior al turneului, BCA ar fi înfrânt de CAB sau de învingătorul lui CAB, deci prin ignorarea lui BCA nu s-a alterat rezultatul final. Când se compară șirul S_i cu șirul S_j , trebuie să se compare $j-i$ elemente (practic subsecvențele care au fost notate cu A și B). La runda k a turneului, șirurile care trebuie comparate vor fi la distanță cel mult 2^k și vor fi $n/2^k$ șiruri de comparat. Deci, numărul total de runde este $\log_2 n$.

25. Se construiește un vector C cu semnificația $C[i] = 0$ dacă nu există nici un subșir care începe pe poziția 1 și se termină pe poziția i sau un număr natural mai mare de 0, care va reprezenta numărul minim de cuvinte în care se poate descompune subșirul care începe pe poziția 1 și se termină pe poziția i . Pentru a calcula acest vector, se folosește următoarea relație de recurență:

$C[i] = \min(C[j] + 1) ; j < i$ și subșirul de la poziția $j+1$ la i apare în dicționar.

Astfel, pentru a calcula $C[i]$ se parcurge dicționarul, se verifică dacă un cuvânt se potrivește astfel încât poziția de sfârșit să fie i și se actualizează $C[i]$. Pentru reconstituire se va mai folosi un alt vector $T[i]$, care va reprezenta indicele cuvântului folosit pentru a obține soluția $C[i]$.

26. Se calculează termenii șirului lui Fibonacci și se transformă numărul dat în baza 10.

27. Se folosește principiul lui Dirichlet: "dacă se pun n obiecte în m cutii ($m < n$) va exista cel puțin o cutie cu cel puțin $\lceil n/m \rceil$ obiecte". ($\lceil \cdot \rceil$ reprezintă rotunjire superioară).

Astfel, se generează numerele: $x, xx, xxx, \dots, xxx\dots xxx$ (de n ori) și se calculează restul la împărțirea cu n . Dacă apare restul 0, s-a găsit o soluție, altfel vor exista două numere cu resturi egale (n numere generate [obiecte] $\rightarrow n-1$ resturi [cutii]), care scăzute vor da un număr de forma $x\dots xx00\dots 0$, multiplu al lui n .

28. Se sortează intervalele după capătul stânga și se atribuie o culoare intervalului în funcție de ce culoare este cea mai puțin "acoperită". La sfârșit, se verifică dacă ambele culori acoperă întregul interval $[1\dots L]$.

29. Fie n , numărul de cifre al numărului B . Se generează toate numerele de lungime cel mult $n/2$ și se formează un palindrom, odată prin concatenarea fiecărui număr cu simetricul său sau prin concatenarea fiecărui număr cu o cifră între 0 și 9 și apoi cu simetricul său. Fiecare palindrom astfel generat se verifică dacă este prim și aparține intervalului $[A, B]$.

30. Se sortează intervalele după capătul dreapta, apoi intervalele care se intersectează se unesc în unul singur, până când nu mai există intervale care să se intersecteze. Primul răspuns va fi cel mai lung interval din cele care există acum, iar al doilea răspuns, cea mai mare "gaură" dintre două intervale adiacente.

31. Se sortează punctele negre după linie și în caz de egalitate, după coloană, pentru a determina câte cuvinte verticale există. Pentru a determina numărul cuvintelor orizontale, se rotește matricea și se aplică același algoritm. Pentru fiecare linie, se parcurg punctele negre de pe linia respectivă și se determină toate secvențele de puncte albe dintre două puncte negre; fiecare astfel de secvență cu lungime mai mare decât 1 se consideră un cuvânt.

32. Se citesc caracterele unul câte unul și se consideră o stivă, inițial goală. De fiecare dată când se întâlnește o paranteză (deschisă sau închisă, dreaptă sau rotundă), aceasta se introduce în stivă. Dacă se întâlnește caracterul "*", atunci se scoate caracterul din vârful stivei (dacă există un astfel de caracter). În final (la întâlnirea caracterului "E"), stiva va conține șirul afișat pe ecran. Pentru a verifica dacă un șir este parantezat corect, acesta se parcurge de la stânga la dreapta și se consideră o a doua stivă, inițial vidă. Dacă se întâlnește o paranteză deschisă dreaptă sau rotundă, aceasta se introduce în vârful stivei. Dacă se întâlnește o paranteză închisă și caracterul din vârful stivei nu este o paranteză deschisă de același tip cu paranteza închisă (dreaptă sau rotundă), atunci se semnalează eroare. Altfel, paranteza deschisă din vârful stivei se elimină și se trece la următorul caracter. Dacă nu s-a semnalat eroare în timpul parcurgerii șirului și la final stiva este vidă, atunci șirul este parantezat corect.

33. Un algoritm destul de eficient este următorul: se sortează punctele după x și în caz de egalitate, după y . Pentru ca dreptunghiul să aibă arie maximă acesta va avea marginile sus și jos la două coordonate y care coincid cu două coordonate ale unor puncte; în caz contrar, dreptunghiul ar putea fi mărit, fără să se introducă puncte în el. Astfel, se iau oricare două coordonate y și se verifică toate dreptunghiurile goale marginite de acestea.

34. Se folosește algoritmul clasic pentru determinarea celui mai lung subșir comun. Toate șirurile se construiesc în paralel cu determinarea lungimilor. Este nevoie de un tablou de mulțimi în care se păstrează toate cele mai lungi subsecvențe comune ale șirurilor (a_1, a_2, \dots, a_i) și (b_1, b_2, \dots, b_j). Există cel mult 1000 de cele mai lungi subsecvențe comune având lungimea maximă egală cu 80 și astfel, dimensiunea tabloului este de 80×80 . Acesta ar necesita 512 MB dacă s-ar folosi memoria statică. Se observă că numai linia curentă și cea precedentă sunt necesare în timpul prelucrării. Astfel, este posibil să se folosească un tablou de dimensiune 2×80 .

35. Fiecărei paranteze magice i se atribuie o paranteză închisă, mai puțin ultimei, căreia i se atribuie câte paranteze închise au mai rămas.

2.1. Subprograme implementate în manieră iterativă

Sectiunea 2.1.1

1. d)	6. b)	11. b)	16. c)
2. b)	7. d)	12. c)	17. c)
3. b),d)	8. a)	13. c)	18. b)
4. c)	9. e)	14. a)	19. a)
5. c)	10. b)	15. a)	20. c)

Sectiunea 2.1.3

```

1. function det(x:integer): integer;
2   var z,y:integer;
3   begin
4     y:=x; z:=x;
5     while not prim(y) do
6       inc(y);
7     while (z>1)and not prim(z)do
8       dec(z);
9     if (z<>1)and(x-z<y-x) then
10      det:=z;
11    else
12      det:=y;
13  end;

int det(int x)
{
  int z,y;
  y=x; z=x;
  while (!prim(y))
    Y++;
  while (z>1 && !prim(z))
    z--;
  if ((z!=1)&&(x-z<y-x))
    return z;
  else
    return y;
}

3. function cub(x:integer):boolean;
4   var i:integer;
5   begin
6     i:=1;
7     while i*i*i<x do inc(i);
8     cub:=i*i*i=x;
9   end;

int cub(int x)
{
  int i;
  i=1;
  while (i*i*i<x) i++;
  return (i*i*i==x);
}

9. function poz(a:sir;n,k:integer):
10   byte;
11   var mx,j,nr,i:integer;
12   begin
13     mx:=0;
14     for i:=1 to n-k+1 do begin
15       nr:=0;
16       for j:=i to i+k-1 do
17         if cub(a[j]) then inc(nr);
18       if nr>mx then begin
19         mx:=nr; poz:=i;
20       end;
21     end;
22   end;

int poz(int a[],int n,int k)
{
  int pz,mx,j,nr,i;
  mx:=0;
  for (i=0;i<n-k+1;i++){
    nr:=0;
    for (j=i;j<=i+k-1;j++)
      if (cub(a[j])) nr++;
    if (nr>mx) {
      mx:=nr; pz=i;
    }
  }
  return pz;
}

```

```

4. function
1   baza(b,x:integer):boolean;
2   var i:integer;
3   begin
4     baza:=true;
5     while x>0 do begin
6       if x mod b>1 then baza:=false;
7       x:=x div b;
8     end;
9   end;

int baza(int b,int x)
{
  int bz;
  while (x>0)
  {
    if (x%b>1) return 0;
    x=x/b;
  }
  return 1;
}

12 procedure det(var a:sir;
13               n,b:integer);
14 var x,i:integer;
15 begin
16   x:=0;
17   i:=0;
18   while i<n do begin
19     if baza(b,x) then begin
20       inc(i);
21       a[i]:=x;
22     end;
23     inc(x);
24   end;
25 end;

void det(int a[],int n,int b)
{
  int x,i;
  x=0;
  i=-1;
  while (i<n-1) {
    if (baza(b,x)) {
      i++;
      a[i]=x;
    }
    x++;
  }
}

5. function cif(x:integer):byte;
2   var i:integer;
3   begin
4     while x>9 do x:=x div 10;
5     cif:=x;
6   end;

int cif(int x)
{
  int i;
  while (x>9) x=x/10;
  return x;
}

8 procedure scrie(a:sir;n:byte);
9   var i,j:byte;
10  ok:boolean;
11  g:text;
12  begin
13    assign(g,'nr.txt');rewrite(g);
14    for i:=0 to 9 do begin
15      ok:=false;
16      for j:=1 to n do
17        if cif(a[j])=i then begin
18          write(g,a[j],');
19          ok:=true;
20        end;
21    if ok then writeln(g);
22  end;
23  close(g);
24 end;

void scrie(int a[],int n)
{
  int ok,i,j;
  FILE *g=fopen("nr.txt","w");
  for (i=1;i<=9;i++){
    ok=0;
    for (j=0;j<n;j++)
      if (cif(a[j])==i){
        fprintf(g,"%d ",a[j]);
        ok=1;
      }
    if (ok) fprintf(g,"\n");
  }
  fclose(g);
}

```

8.

```

1 function rest(a:sir;n,k:byte):byte;
2 var s,i:integer;
3 begin
4   s:=0;
5   for i:=1 to k do
6     s:=(s+a[i])mod n;
7   rest:=s;
8 end;
9 procedure det(a:sir;n:byte;
10               var x,y:byte);
11 var z,i,j:byte;
12   r:array[0..100]of byte;
13 begin
14   for i:=0 to 100 do r[i]:=0;
15   for i:=1 to n do begin
16     z:=rest(a,n,i);
17     if z=0 then begin
18       x:=1; y:=i;
19     end
20     else
21       if r[z]=0 then r[z]:=i
22       else begin
23         x:=r[z]+1; y:=i;
24       end;
25   end;
26 end;

```

10.

```

1 function pflix(a:sir;n:byte):
2   boolean;
3 var s,i:integer;
4 begin
5   pflix:=false;
6   for i:=1 to n do
7     if a[i]=i then pflix:=true;
8   end;
9 procedure perm(var a:sir;n:byte);
10 var i:byte;
11 begin
12   a[n+1]:=a[1];
13   for i:=1 to n do a[i]:=a[i+1];
14 end;
15 procedure scribe(a:sir;n:byte);
16 var j,i:byte;
17 begin
18   for i:=1 to n do begin
19     perm(a,n);
20     if not pflix(a,n) then begin
21       for j:=1 to n do
22         write(a[j], ' ');
23       writeln;
24     end;
25   end;
26 end;

```

```

int rest(int a[],int n,int k)
{int s,i;
 s=0;
 for (i=0;i<=k;i++)
   s=(s+a[i])%n;
 return s;
}
void det(int a[],int n,int
&x,int &y)
{
 int z,i,j,r[100];
 for (i=0;i<100;i++)
   r[i]=-1;
 for (i=0;i<n;i++)
 {
  z=rest(a,n,i);
  if (z==0){
    x=0; y=i;
  }
  else
    if (r[z]==-1) r[z]=i;
    else {
      x=r[z]+1; y=i;
    }
 }
}

```

```

int pflix(int a[],int n)
{
 int i;
 for (i=0;i<n;i++)
   if (a[i]==i+1) return 1;
 return 0;
}
void perm(int a[],int n)
{
 int i;
 a[n]=a[0];
 for (i=0;i<n;i++)
   a[i]=a[i+1];
}
void scribe(int a[],int n)
{
 int j,i;
 for (i=0;i<n;i++){
   perm(a,n);
   if (!pflix(a,n)){
     for (j=0;j<n;j++)
       cout<<a[j]<<' ';
     cout<<endl;
   }
 }
}

```

11.

```

1 function suml(a:mat;n,l:byte):
2   integer;
3 var s,i:integer;
4 begin
5   s:=0;
6   for i:=1 to n do s:=s+a[l,i];
7   suml:=s;
8 end;
9
10 function sumc(a:mat;n,c:byte):
11   integer;
12 var s,i:integer;
13 begin
14   s:=0;
15   for i:=1 to n do s:=s+a[i,c];
16   sumc:=s;
17 end;
18
19 procedure scribe (a:mat;n:byte);
20 var j,i:byte;x:integer;
21 begin
22   for i:=1 to n do begin
23     x:=suml(a,n,i);
24     for j:=1 to n do
25       if x=sumc(a,n,j) then
26         write(a[i,j], ' ');
27   end;
28 end;

```

```

int suml(int a[10][10],int n,
int l)
{
 int s,i;
 s=0;
 for (i=0;i<n;i++) s+=a[l][i];
 return s;
}
int sumc(int a[10][10],int n,
int c)
{
 int s,i;
 s=0;
 for (i=0;i<n;i++) s+=a[i][c];
 return s;
}
void scribe (int a[10][10],int n)
{
 int j,i,x;
 for (i=0;i<n;i++){
   x=suml(a,n,i);
   for (j=0;j<n;j++)
     if (x==sumc(a,n,j))
       cout<<a[i][j]<<' ';
 }
}

```

13.

```

1 function nr(x,y,c:byte):byte;
2 var s,i:integer;
3 begin
4   s:=0;
5   for i:=1 to m do
6     if (a[i,c]<x)or(a[i,c]>y) then
7       inc(s);
8   nr:=s;
9 end;
10
11 procedure det(x,y:integer);
12 var j,i,max,z:integer;
13 begin
14   max:=0;
15   for i:=1 to m do begin
16     z:=nr(x,y,i); a[n+1,i]:=z;
17     if z>max then max:=z;
18   end;
19   for i:=1 to m do
20     if a[n+1,i]=max then
21       write(i, ' ');
22 end;

```

```

int nr(int x,int y,int c)
{
 int s,i;
 s=0;
 for (i=0;i<m;i++)
   if (a[i][c]<x||a[i][c]>y)
     s++;
 return s;
}
void det(int x,int y)
{
 int j,i,max,z;
 max=0;
 for (i=0;i<m;i++){
   z=nr(x,y,i); a[n][i]=z;
   if (z>max) max=z;
 }
 for (i=0;i<m;i++)
   if (a[n][i]==max)
     cout<<i+1<<' ';
}

```

16.

```

1 procedure perm(var a:mat;
2                 n,l:byte);
3 var s,i:integer;
4 begin
5   a[l,n+1]:=a[l,1];
6   for i:=1 to n do
7     a[l,i]:=a[l,i+1];
8   end;
9
10 procedure solve(var a:mat;
11                 n,k:byte);
12 var j,i:integer;
13 begin
14   for i:=1 to n do
15     for j:=1 to k do perm(a,n,i)
16   end;

```

17.

```

1 function exp(x,p:word):byte;
2 var y:integer;
3 begin
4   y:=0;
5   while x mod p=0 do begin
6     inc(y);
7     x:=x div p;
8   end;
9   exp:=y;
10 end;
11
12 function nrz(a:mat;m,l:byte):
13             byte;
14 var n2,n5,i:byte;
15 begin
16   n2:=0; n5:=0;
17   for i:=1 to m do begin
18     n2:=n2+exp(a[l,i],2);
19     n5:=n5+exp(a[l,i],5);
20   end;
21   if n2<n5 then nrz:=n2
22   else nrz:=n5;
23 end;
24
25 procedure scribe(a:mat;n:byte);
26 var j,i:integer;
27 begin
28   for i:=1 to n do
29     write(nrz(a,n,i),' ');
30 end;

```

18.

```

1 function baza:byte;
2 var bm,i,j:byte;
3 begin
4   bm:=a[1,1];

```

```

void perm(int a[10][10],
          int n,int l)
{
  int s,i;
  a[l][n]=a[l][0];
  for (i=0;i<n;i++)
    a[l][i]=a[l][i+1];
}

void solve(int a[10][10],
           int n,int k)
{
  int j,i;
  for (i=0;i<n;i++)
    for (j=1;j<=k;j++)
      perm(a,n,i);
}

```

```

int exp(long x,long p)
{
  int y;
  y=0;
  while (x%p==0){
    y++; x/=p;
  }
  return y;
}

int nrz(long a[10][10],int m,
        int l)
{
  int n2,n5,i;
  n2=0; n5=0;
  for (i=0;i<m;i++){
    n2+=exp(a[l][i],2);
    n5+=exp(a[l][i],5);
  }
  if (n2<n5) return n2;
  else return n5;
}

void scribe(long a[10][10],int n)
{
  int j,i;
  for (i=0;i<n;i++)
    cout<<nrz(a,n,i)<<' ';
}

```

```

int baza()
{
  int bm,i,j;
  bm=a[0][0];
}

```

5

```

for i:=1 to n do
  for j:=1 to m do
    if a[i,j]>bm then
      bm:=a[i,j];
  baza:=bm+1;
end;

function conv(b,l:byte):
                longint;
var i,x:longint;
begin
  x:=0;
  for i:=1 to m do
    x:=x*b+a[l,i];
  conv:=x;
end;

procedure scribe;
var j,i:integer;
begin
  j:=baza;
  for i:=1 to n do
    write(conv(j,i),' ');
end;

```

19.

```

function eval(l:byte;s:string):
                longint;
var x,i,j:byte;
begin
  x:=a[l,1];
  for i:=2 to m do
    if s[i-1]='+' then
      x:=x+a[l,i]
    else x:=x-a[l,i];
  eval:=x;
end;

function max:longint;
var i,mx:integer;
begin
  mx:=eval(1,s);
  for i:=2 to n do
    if mx<eval(i,s) then
      mx:=eval(i,s);
  max:=mx;
end;

```

```

for (i=0;i<n;i++)
  for (j=0;j<m;j++)
    if (a[i][j]>bm)
      bm=a[i][j];
return bm+1;
}

long conv(int b,int l)
{
  int i,x;
  x=0;
  for (i=0;i<m;i++)
    x=x*b+a[l][i];
  return x;
}

void scribe()
{
  int j,i;
  j=baza();
  for (i=0;i<n;i++)
    cout<<conv(j,i)<<' ';
}

```

```

long eval(int l, char *s)
{
  int x,i,j;
  x=a[l][0];
  for (i=1;i<m;i++)
    if (s[i-1]=='+')
      x=x+a[l][i];
    else x=x-a[l][i];
  return x;
}

long max()
{
  int i,mx;
  mx=eval(0,s);
  for (i=1;i<n;i++)
    if (mx<eval(i,s))
      mx=eval(i,s);
  return mx;
}

```


2.2. Subprograme implementate în manieră recursivă

Sectiunea 2.2.1

1. b)	7. a)	13. b)	20. c)
2. a)	8. b)	14. c)d)	21. a)
3. c)	9. d)	15. c)	22. b)c)
4. d)	10. c)	16. d)	23. b)
5. c)	11. b)	17. c)	24. a)
6. c)	12. a)b)d)	18. c)d)	25. b)
		19. d)	26. d)

Sectiunea 2.2.3

```

1. function ok(x,i:longint):
   boolean;
2. begin
3.   if x=1 then ok:=true
4.   else
5.     if x mod i<>0 then ok:=false
6.     else ok:=ok(x div i,i+1)
7.   end;
8. end;
9.
10. function nr(i:integer):
   integer;
11. var x:longint;
12. begin
13.   if i<=n then begin
14.     readln(x);
15.     if ok(x,2) then
16.       nr:=nr(i+1)+1
17.     else nr:=nr(i+1);
18.   end
19.   else nr:=0;
20. end;
21.

```

```

2. function cif(x:longint):byte;
3. begin
4.   if x<10 then cif:=x
5.   else
6.     cif:=cif(x div 10)
7.   end;
8. procedure afis(c,i:byte);
9. var x:longint;
10. begin
11.   if i<=n then begin
12.     if cif(a[i])=c then
13.       write(a[i],' ');
14.     afis(c,i+1);
15.   end;
16. end;

```

```

int ok(long x,long i)
{
  if (x==1) return 1;
  else
    if (x%i) return 0;
    else return ok(x/i,i+1);
}

int nr(int i)
{
  long x;
  if (i<=n){
    cin>>x;
    if (ok(x,2))
      return nr(i+1)+1;
    else
      return nr(i+1);
  }
  else return 0;
}

```

```

int cif(long x)
{
  if (x<10) return x;
  else
    return cif(x/10);
}

void afis(int c,int i)
{
  long x;
  if (i<=n) {
    if (cif(a[i])==c)
      cout<<a[i]<<' ';
    afis(c,i+1);
  }
}

```

```

3. function nr(x:longint):byte;
1. begin
2.   if x=0 then nr:=0
3.   else
4.     if x mod 2=0 then
5.       nr:=nr(x div 10)+1
6.     else nr:=nr(x div 10)
7.   end;
8. end;
9.
10. procedure sortare(i,j:byte);
11. var x:longint;
12. begin
13.   if i<n then
14.     if j<=n then begin
15.       if (nr(a[i])>nr(a[j]))or
16.         (nr(a[i])=nr(a[j]))and
17.         (a[i]>a[j]) then
18.         begin
19.           x:=a[i];a[i]:=a[j];
20.           a[j]:=x;
21.         end;
22.       sortare(i,j+1)
23.     end
24.     else sortare(i+1,i+2)
25.   end;

```

```

4. function p(x:longint):longint;
1. begin
2.   if x=0 then p:=1
3.   else
4.     if x mod 2=1 then
5.       p:=p(x div 10)*x mod 10
6.     else p:=p(x div 10)
7.   end;
8. end;
9.
10. procedure del(var a:sir;
11.   i:byte;var n:byte);
12. var x:longint;
13. begin
14.   if i<n then begin
15.     a[i]:=a[i+1];
16.     del(a,i+1,n)
17.   end else n:=n-1;
18. end;

```

```

6. function cmax(x:longint):byte;
1. begin
2.   if x=0 then cmax:=0
3.   else
4.     if x mod 10>cmax(x div 10)
5.     then cmax:=x mod 10
6.     else cmax:=cmax(x div 10)
7.   end;
8. end;

```

```

int nr(long x)
{
  if (x==0) return 0;
  else
    if (x%2==0)
      return nr(x/10)+1;
    else return nr(x/10);
}

void sortare(int i,int j)
{
  long x;
  if (i<n-1)
    if (j<n){
      if (nr(a[i])>nr(a[j]))||
        (nr(a[i])==nr(a[j]))&&
        a[i]>a[j])
      {
        x=a[i];a[i]=a[j];
        a[j]=x;
      }
      sortare(i,j+1);
    }
    else sortare(i+1,i+2);
}

```

```

long p(long x)
{
  if (x==0) return 1;
  else
    if (x%2==1)
      return p(x/10)*x%10;
    else return p(x/10);
}

void del(long a[],int i,
  int &n)
{
  long x;
  if (i<n-1){
    a[i]=a[i+1];
    del(a,i+1,n);
  }
  else n--;
}

```

```

int cmax(long x)
{
  if (x==0) return 0;
  else
    if (x%10>cmax(x/10))
      return x%10;
    else return cmax(x/10);
}

```

```

9 function delcif(x,c:longint):
10     longint;
11 begin
12     if x=0 then delcif:=0
13     else
14         if x mod 10=c then
15             delcif:=delcif(x div 10,c)
16         else
17             delcif:=10*delcif(x div 10,c)+
18                 x mod 10;
19 end;
20
21 procedure s(var a:sir;i,n:byte);
22 var x:longint;
23 begin
24     if i<=n then begin
25         a[i]:=delcif(a[i],cmax(a[i]));
26         s(a,i+1,n)
27     end
28 end;

```

```

7.
1 function pal(x,y:integer;var
2   z:integer):boolean;
3 begin
4     if x=0 then pal:=y=z
5     else begin
6         z:=z*10+x mod 10;
7         pal:=pal(x div 10,y,z)
8     end;
9 end;
10
11 function search(x,t:integer):
12     integer;
13 var z:integer;
14 begin
15     z:=0;
16     if pal(x,x,z) then search:=x
17     else search:=search(x+t,t);
18 end;
19
20 procedure s(var a:sir;i,n:byte);
21 begin
22     if i<=n then begin
23         if (search(a[i],1)-a[i] <
24             a[i]-search(a[i],-1))
25         then a[i]:=search(a[i],1)
26         else a[i]:=search(a[i],-1);
27         s(a,i+1,n)
28     end
29 end;

```

```

long delcif(long x,int c)
{
    if (x==0) return 0;
    else
        if (x%10==c)
            return delcif(x/10,c);
        else
            return
                10*delcif(x/10,c)+x%10;
}

void s(long a[],int i,int n)
{
    long x;
    if (i<n)
    {
        a[i]=delcif(a[i],cmax(a[i]));
        s(a,i+1,n);
    }
}

```

```

int pal(int x,int y,int &z).
{
    if (x==0) return (y==z);
    else {
        z=z*10+x%10;
        return pal(x/10,y,z);
    }
}

int search(int x,int t)
{
    int z;
    z=0;
    if (pal(x,x,z)) return x;
    else
        return search(x+t,t);
}

void s(int a[],int i,int n)
{
    if (i<n) {
        if (search(a[i],1)-a[i]<
            a[i]-search(a[i],-1))
            a[i]=search(a[i],1);
        else
            a[i]=search(a[i],-1);
        s(a,i+1,n);
    }
}

```

```

10.
1 function nr(i,t:integer):
2     longint;
3 begin
4     if (i=0) or (i>n) then nr:=0
5     else nr:=10*nr(i+t,t)+a[i]
6     end;
7
8 function cub(x,i:longint):
9     boolean;
10 begin
11     if i*i=i=x then cub:=true
12     else if i*i>x then
13         cub:=false
14         else cub:=cub(x,i+1);
15 end;

```

```

11.
1 function inv(x:longint;
2     var y:longint):longint;
3 begin
4     if x=0 then inv:=y
5     else begin
6         y:=y*10+x mod 10;
7         inv:=inv(x div 10,y);
8     end;
9 end;
10
11 procedure S(i,x:longint);
12 var y,z:longint;
13 begin
14     if i<x then begin
15         y:=0; z:=inv(i,y);
16         if (i+z=x) and (i mod 10<>0)
17         then writeln(i,' ',z);
18         S(i+1,x);
19     end;
20 end;

```

```

12.
1 function Sd(x,i:word):word;
2 begin
3     if i>x div 2 then Sd:=0
4     else
5         if x mod i=0 then
6             Sd:=Sd(x,i+1)+i
7         else Sd:=Sd(x,i+1);
8     end;
9
10 procedure Make(i:byte;
11     var b,d:sir;var k,l:byte);
12 begin
13     if i<=n then begin
14         if Sd(a[i],1)=a[i] then

```

```

long nr(int i,int t)
{
    if (i==0 || i==n) return 0;
    else
        return 10*nr(i+t,t)+a[i];
}

int cub(long x,long i)
{
    if (i*i==x) return 1;
    else if (i*i>x) return 0;
    else
        return cub(x,i+1);
}

```

```

long inv(long x,long &y)
{
    if (x==0) return y;
    else {
        y=y*10+x%10;
        return inv(x/10,y);
    }
}

void S(long i,long x)
{
    long y,z;
    if (i<x){
        y=0; z=inv(i,y);
        if (i+z==x && i%10)
            cout<<i<<' '<<z<<endl;
        S(i+1,x);
    }
}

```

```

long Sd(long x,long i)
{
    if (i>x/2) return 0;
    else
        if (x%i==0)
            return Sd(x,i+1)+i;
        else return Sd(x,i+1);
}

void Make(int i,long b[],
    long d[],int &k,int &l)
{
    if (i<n){
        if (Sd(a[i],1)==a[i])

```

```

15 begin
16   inc(k);
17   b[k]:=a[i];
18 end
19 else begin
20   inc(l);
21   d[l]:=a[i];
22 end;
23 Make(i+1,b,d,k,l);
24 end;
25 end;

```

16.

```

1 function fact(x:byte):longint;
2 begin
3   if x=0 then fact:=1
4   else fact:=x*fact(x-1)
5 end;
6
7 function sum(x:integer):longint;
8 begin
9   if x=0 then sum:=0
10  else sum:=fact(x mod 10)+
11    sum(x div 10)
12 end;
13
14 function prod(i:integer):longint;
15 var x:integer;
16 begin
17   if i>n then prod:=1
18   else begin
19     readln(x);
20     if sum(x)=x then
21       prod:=prod(i+1)*x
22     else
23       prod:=prod(i+1)
24   end;

```

17.

```

1 function T(x:longint):longint;
2 begin
3   if x=1 then T:=0
4   else T:=x-T(x-1);
5 end;
6
7 procedure M(i,j:byte;
8   var a:mat);
9 begin
10  if i<=n then
11    if j<=n then begin
12      a[i,j]:=T(n*(i-1)+j);
13      M(i,j+1,a);
14    end
15  else M(i+1,1,a);
16 end;

```

```

{
  k++;
  b[k]:=a[i];
}
else {
  l++;
  d[l]:=a[i];
}
Make(i+1,b,d,k,l);
}
}

```

```

long fact(int x)
{
  if (x==0) return 1;
  else return x*fact(x-1);
}

long sum(int x)
{
  if (x==0) return 0;
  else return
    fact(x%10)+sum(x/10);
}

long prod(int i)
{int x;
  if (i>n) return 1;
  else {
    cin>>x;
    if (sum(x)==x)
      return prod(i+1)*x;
    else
      return prod(i+1);
  }
}

```

```

long T(long x)
{
  if (x==1) return 0;
  else return x-T(x-1);
}

void M(int i,int j,
  int a[10][10])
{
  if (i<n)
    if (j<n) {
      a[i][j]:=T(n*i+j+1);
      M(i,j+1,a);
    }
  else M(i+1,0,a);
}

```

18.

```

1 function ok(x,i:longint):
2   boolean;
3 begin
4   if i>x div 2 then ok:=true
5   else
6     if x mod i=0 then ok:=false
7     else ok:=ok(x,i+1)
8   end;
9 procedure inv(l,i:byte);
10 var x:integer;
11 begin
12   if i<=m div 2 then begin
13     x:=a[l,i];
14     a[l,i]:=a[l,m-i+1];
15     a[l,m-i+1]:=x;
16     inv(l,i+1);
17   end;
18 end;
19 procedure pl(i:byte);
20 begin
21   if i<=n then begin
22     if ok(a[i,1],2) then
23       inv(i,1);
24     pl(i+1);
25   end;
26 end;

```

19.

```

1 function S1(i,j:byte):integer;
2 begin
3   if i<=(n+1) div 2 then
4     if j<=n-i+1 then
5       S1:=S1(i,j+1)+a[i,j]
6     else S1:=S1(i+1,i+1)
7     else S1:=0;
8   end;
9
10 function S2(i,j:byte):integer;
11 begin
12   if i<=n then
13     if j<=i then
14       S2:=S2(i,j+1)+a[i,j]
15     else S2:=S2(i+1,n-i)
16     else S2:=0;
17   end;
18
19 function S3(j,i:byte):integer;
20 begin
21   if j<=(n+1) div 2 then
22     if i<=n-j+1 then
23       S3:=S3(j,i+1)+a[i,j]
24     else S3:=S3(j+1,j+1)
25     else S3:=0;
26 end;

```

```

int ok(int x,int i)
{
  if (i>x/2) return 1;
  else
    if (x%i==0) return 0;
    else return ok(x,i+1);
}

void inv(int l,int i)
{
  int x;
  if (i<=m/2) {
    x=a[l][i];
    a[l][i]=a[l][m-i-1];
    a[l][m-i-1]=x;
    inv(l,i+1);
  }
}

void pl(int i)
{
  if (i<n)
  {
    if (ok(a[i][0],2))
      inv(i,0);
    pl(i+1);
  }
}

```

```

int S1(int i,int j)
{
  if (i<=n/2)
    if (j<=n-i-1)
      return S1(i,j+1)+a[i][j];
    else return S1(i+1,i+1);
  else return 0;
}

```

```

int S2(int i,int j)
{
  if (i<n)
    if (j<=i)
      return S2(i,j+1)+a[i][j];
    else return S2(i+1,n-i-2);
  else return 0;
}

```

```

int S3(int j,int i)
{
  if (j<=n/2)
    if (i<=n-j-1)
      return S3(j,i+1)+a[i][j];
    else return S3(j+1,j+1);
  else return 0;
}

```

```

27. function S4(j,i:byte):integer;
28. begin
29.   if j<=n then
30.     if i<=j then
31.       S4:=S4(j,i+1)+a[i,j]
32.     else S4:=S4(j+1,n-j)
33.     else S4:=0;
34. end;

```

```

20.
1. procedure OrdL(i,j,l:byte);
2. var x:integer;
3. begin
4.   if i<m then
5.     if j<=m then begin
6.       if a[l,i]>a[l,j] then begin
7.         x:=a[l,i]; a[l,i]:=a[l,j];
8.         a[l,j]:=x;
9.       end;
10.      OrdL(i,j+1,l);
11.    end
12.    else OrdL(i+1,i+2,l);
13.  end;
14. procedure OrdC(i,j,c:byte);
15. var x:integer;
16. begin
17.   if i<n then
18.     if j<=n then begin
19.       if a[i,c]>a[j,c] then begin
20.         x:=a[i,c]; a[i,c]:=a[j,c];
21.         a[j,c]:=x;
22.       end;
23.       OrdC(i,j+1,c);
24.     end
25.     else OrdC(i+1,i+2,c);
26.  end;

```

```

21.
1. function S_e(x,i:integer):byte;
2. begin
3.   if x=1 then S_e:=0
4.   else
5.     if x mod i=0 then
6.       S_e:=1+S_e(x div i,i)
7.     else S_e:=S_e(x,i+1)
8.   end;
9. procedure Diag(i,j:byte);
10. var x:integer;
11. begin
12.   if i<=n then
13.     if j<=n then begin
14.       a[i,j]:=S_e(a[i,j],2);
15.       Diag(i,j+1)
16.     end
17.     else Diag(i+1,n-i+1);
18.  end;

```

```

int S4(int j,int i)
{
  if (j<n)
    if (i<=j)
      return S4(j,i+1)+a[i][j];
    else return S4(j+1,n-j-2);
  else return 0;
}

```

```

void OrdL(int i,int j,int l)
{int x;
  if (i<m-1)
    if (j<m)
      {
        if (a[l][i]>a[l][j]) {
          x=a[l][i]; a[l][i]=a[l][j];
          a[l][j]=x;
        }
        OrdL(i,j+1,l);
      }
    else OrdL(i+1,i+2,l);
}

```

```

void OrdC(int i,int j,int c)
{int x;
  if (i<n-1)
    if (j<n)
      {
        if (a[i][c]>a[j][c]) {
          x=a[i][c]; a[i][c]=a[j][c];
          a[j][c]=x;
        }
        OrdC(i,j+1,c);
      }
    else OrdC(i+1,i+2,c);
}

```

```

int S_e(int x,int i)
{
  if (x==1) return 0;
  else
    if (x%i==0)
      return 1+S_e(x/i,i);
    else
      return S_e(x,i+1);
}

void Diag(int i,int j)
{int x;
  if (i<n)
    if (j<n)
      { a[i][j]:=S_e(a[i][j],2);
        Diag(i,j+1);
      }
    else Diag(i+1,n-i-1);
}

```

```

22.
1. function T(x,i:longint):boolean;
2. begin
3.   if i>x div 2 then T:=true
4.   else
5.     if x mod i=0 then T:=false
6.     else T:=T(x,i+1);
7.   end;
8. procedure M(i,j,v:byte);
9. begin
10.  if i<=n then
11.    if j<=n then begin
12.      if T(v,2) then begin
13.        a[i,j]:=v;
14.        M(i,j+1,v+1); end
15.      else M(i,j,v+1);
16.    end
17.    else M(i+1,1,v+1);
18.  end;

```

```

23.
1. procedure Morse(s:string);
2. begin
3.   if length(s)=n then
4.     writeln(s)
5.   else begin
6.     Morse(s+'-');
7.     Morse(s+'.'');
8.   end;
9. end;

```

```

int T(long x,long i)
{
  if (i>x/2) return 1;
  else
    if (x%i==0) return 0;
    else return T(x,i+1);
}

void M(int i,int j,int v)
{
  if (i<n)
    if (j<n) {
      if (T(v,2)) {
        a[i][j]=v;
        M(i,j+1,v+1);
      }
      else M(i,j,v+1);
    }
    else M(i+1,0,v+1);
}

```

```

void Morse(char s[]) {
  char x[256];
  if (strlen(s)==n)
    cout<<s<<endl;
  else {
    strcpy(x,s); x[strlen(x)]='-';
    Morse(x);
    strcpy(x,s); x[strlen(x)]='.';
    Morse(x);
  }
}

```

Secțiunea 2.3.2

1. Pornind de la sfârșitul șirului de numere, se va construi un tablou A , în care A_i reprezintă numărul de subșiruri strict crescătoare de lungime maximă care încep cu elementul din poziția i . Acest tablou se construiește folosind rezolvarea problemei subșirului strict crescător de lungime maximă, deci se va construi în paralel un tablou L , unde L_i reprezintă lungimea celui mai lung subșir strict crescător care începe pe poziția i . Fie MAX valoarea maximă din L . Suma elementelor din A , corespunzătoare elementelor maxime din L , reprezintă numărul total de subșiruri strict crescătoare de lungime maximă. Ne interesează al K -lea astfel de subșir. Fie $i1$ poziția primei apariții a lui MAX în L , $i2$ poziția celei de-a doua apariții, ș.a.m.d. Dacă $A_{i1} \geq K$, atunci subșirul căutat este al K -lea subșir strict crescător maximal care începe pe poziția $i1$. Dacă $A_{i1} \leq K \leq A_{i1} + A_{i2}$, atunci subșirul căutat începe cu poziția $i2$, etc. După ce a fost găsit primul element al subșirului căutat, se determină al doilea element, apoi al treilea, etc. Metoda este similară. Se impune următoarea observație: dacă, de exemplu, subșirul începe cu poziția $i3$, se caută al $(K - A_{i1} + A_{i2})$ -lea subșir crescător maximal care începe după poziția $i3$, are lungimea $MAX-1$, iar primul element al său este mai mare decât elementul de pe poziția $i3$. Acest subșir se caută folosind același procedeu.

2. Se calculează, pentru fiecare element, lungimea celui mai lung subșir strict crescător care se termină cu el și lungimea celui mai lung subșir strict descrescător care începe cu el (vezi soluția de la problema precedentă pentru detalii). Soluția constă în păstrarea a două astfel de subșiruri de războinici (unul crescător și unul descrescător) pentru doi războinici de aceeași înălțime (eventual identici) și eliminarea celorlalți. Războinicii din primul subșir privesc spre stânga, ceilalți spre dreapta. Primul subșir se termină înainte de a începe al doilea. Se au în vedere cazurile particulare.

3. Un număr are aspect de munte dacă pe primele poziții se află cifre în ordine crescătoare, iar pe ultimele poziții se află cifre în ordine descrescătoare. Pentru a rezolva problema se va determina, pentru fiecare cifră în parte, lungimea celui mai lung subșir crescător care se termină cu cifra respectivă, precum și lungimea celui mai lung subșir descrescător care începe în dreptul cifrei respective. Pentru fiecare cifră se va calcula sumele celor două lungimi, iar "vârful" muntelui va fi dat de cifra pentru care suma este maximă. Soluția problemei va fi dată de diferența dintre numărul de cifre al numărului dat și suma respectivă, la care se va adăuga valoarea 1. Valoarea 1 trebuie adăugată deoarece "vârful" muntelui este considerat de două ori: în șirul crescător și în șirul descrescător.

4. Se fixează ordinea în care vor fi aranjate culorile în mână (sunt $C!$ variante), iar pentru fiecare ordine se determină numărul minim de cărți care se vor muta. Pentru a genera toate posibilitățile de ordonare, se poate considera fiecare număr între 1 și $C!$, fiecare reprezentând indicele unei permutări, dacă acestea ar fi sortate lexicografic; decodificarea numărului se face asemănător problemei complementare (vezi capitolul 1.3.1, problema Permutări). Pentru a determina acest număr, se găsește cel mai lung subșir crescător (ținând cont și de ordinea culorilor aleasă), rezultatul fiind numărul de elemente – lungimea celui mai lung subșir crescător. Se va alege cel mai bun rezultat.

5. Se observă că pentru primul grup va trebui aleasă o submulțime a numerelor divizibilă cu valoarea N . Pentru al doilea grup, dintre cei k soldați rămași, va trebui să se aleagă o submulțime divizibilă cu valoarea k . Așadar, pentru al doilea grup este aceeași problemă pentru o valoare k strict mai mică decât n . Practic, la fiecare pas se va rezolva aceeași problemă pentru valori din ce în ce mai mici până în momentul în care nu mai există nici un număr. Așadar, la fiecare pas va trebui să se determine o submulțime a unei mulțimi cu k elemente a cărei sumă să fie divizibilă cu k . În acest scop se va folosi principiul cutiei lui Dirichlet. Se vor calcula sumele $S_i = a_1 + a_2 + \dots + a_i$ folosind formula recurentă $S_i = a_i + S_{i-1}$. Potrivit principiului cutiei lui Dirichlet (vezi capitolul 1.3), dacă resturile împărțirii valorilor S_i la k sunt distincte, atunci unul dintre aceste resturi este 0. Potrivit aceluiași principiu, dacă resturile nu sunt distincte, atunci două dintre acestea sunt egale. Dacă resturile împărțirii valorilor S_x și S_y ($x < y$) sunt egale, atunci restul împărțirii valorii $S_y - S_x$ la k este 0. Așadar, valoarea $a_{x+1} + a_{x+2} + \dots + a_y$ va fi divizibilă cu k . În concluzie, va exista întotdeauna posibilitatea de a alege o submulțime a cărei sumă să fie divizibilă cu k . După alegerea acestei submulțimi, dacă ea nu conține toate cele k

elemente, va trebui să se rezolve aceeași problemă pentru o valoare strict mai mică decât k (elementele mulțimii respective sunt eliminate).

6. Această problemă poate fi ușor rezolvată folosind principiul cutiei lui Dirichlet. Se va concatena numărul format din cifrele date cu el însuși până în momentul în care restul împărțirii la n va fi 0 sau se obține un rest care a mai fost obținut la un pas anterior. Pe baza principiului, sigur se va ajunge în una dintre cele două situații după cel mult n pași. Dacă s-a obținut restul 0, s-a găsit un rezultat. Dacă nu, rezultatul va fi dat prin simpla scădere a celor două numere care duc la obținerea aceluiași rest prin împărțirea la n . Dacă cele două resturi se obțin la iterațiile p și q , atunci diferența constă în simpla înlocuire a ultimelor $k^*(q-p)$ cifre ale rezultatului cu cifra 0 (care face parte cu siguranță dintre cifrele date). Rezultatul nu poate să înceapă cu cifra 0. Pentru a evita această situație este suficient ca, în șirul cifrelor date, să nu se pună cifra 0 pe prima poziție.

7. Dacă n este un număr natural și a este un șir care conține puterile la care apar factorii primi din descompunerea lui n , atunci numărul divizorilor lui n este:

$$\sum_{i=0}^k (a_i + 1)$$

Demonstrația acestei teoreme este foarte simplă. Numărului n i se asociază șirul a_i al puterilor la care apar factorii primi din descompunerea sa. Astfel, fiecărui divizor îi va corespunde un element al produsului cartezian $X_1 * X_2 * \dots * X_k$, unde $X_i = \{0, 1, \dots, a_i\}$, iar numărul elementelor acestui produs cartezian este exact suma de mai sus. De fapt, nu trebuie să luate în considerare toate numerele prime, fiind suficiente doar cele mai mici. În continuare, se va presupune că s-au ales primele 15 numere prime. La început, se determină vectorul d al divizorilor numărului n ; se consideră că acest vector are m elemente. Se construiește o matrice A cu 15 linii și m coloane, $A_{i,j}$ având valoarea egală cu cel mai mic număr care are ca factori primi primele i numere prime și având d_j divizori. Rezultatul cerut va fi valoarea minimă de pe ultima coloană a matricei. Primul element al fiecărei linii ($A_{i,1}$) va avea valoarea 1, deoarece 1 este cel mai mic număr cu un singur divizor; $A_{0,i} = \infty$, deoarece nu există numere cu i divizori și nici un factor prim. Elementul $A_{i,j}$ poate fi calculat folosind valorile $A_{i-1,k}$, pentru $k < j$ și d_k este divizor al lui d_j . $A_{i-1,k}$ are d_k divizori, deci o valoare $A_{i,j}$ cu d_j divizori poate fi obținută calculând valoarea:

$$A_{i-1,k} \cdot p_i^{d_j/d_k+1}$$

unde p_i este al i -lea număr prim. Valoarea $A_{i,j}$ va fi aleasă ca fiind minimul acestor valori. Dacă această valoare minimă este mai mare decât $A_{i-1,j}$, atunci $A_{i,j}$ va primi valoarea $A_{i-1,j}$. Deoarece ar trebui să se lucreze cu numere foarte mari, în program se va determina logaritmul natural al valorilor $A_{i,j}$, iar în final calculăm valoarea reală a minimului de pe ultima coloană a matricei folosind operații cu numere mari.

8. Această problemă poate fi rezolvată foarte ușor păstrând un șir de valori booleene care să indice dacă o anumită sumă poate sau nu fi obținută. Datorită faptului că sunt cel mult 100 de numere ale căror valori sunt cel mult 1000, suma maximă care poate fi obținută este 100.000, deci ar fi nevoie de 100.001 valori.

Inițial, se va considera că nu poate fi obținută decât suma 0. În continuare, la fiecare pas, se va lua în considerare un număr. Se va parcurge șirul sumelor obținute la pasul anterior și pentru fiecare poziție cu valoarea 1, la suma corespunzătoare se va adăuga numărul curent. Rezultatul operației va reprezenta o sumă care poate fi obținută cu siguranță. Se observă că, dacă se parcurge șirul de la stânga la dreapta (începând cu valoarea 0), în momentul în care valoarea corespunzătoare unei sume devine 1, dacă aceasta este setată în același șir, atunci când se va ajunge la poziția respectivă nu se va putea ști dacă valoarea a fost setată la pasul curent sau la un pas anterior. Problema amintită este evitată prin simpla parcurgere a șirului de la dreapta spre stânga (dinspre suma maximă spre 0). Fiecare valoare nou-setată se va afla la dreapta poziției curente, deci nu va mai fi întâlnită la pasul curent datorită deplasării spre stânga.

9. Se implementează o procedură de înmulțire a unui număr mare cu un număr mic și se aplică următoarea definiție recursivă (care va realiza un număr logaritm de înmulțiri):

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2}, & n \text{ par} \\ a^{n-1} \cdot a, & n \text{ impar} \end{cases}$$

10. Se folosește următoarea formulă pentru calcularea ariei unui triunghi determinat de trei puncte ale căror coordonate sunt întregi:

$$\frac{(x_2 - x_1) \cdot (y_3 - y_1) - (x_3 - x_1) \cdot (y_2 - y_1)}{2}$$

De asemenea, se observă că paritatea ariei este determinată de paritatea celor șase coordonate. Din acest motiv, se vor împărți punctele în patru categorii (ambele coordonate pare, ambele coordonate impare, prima coordonată pară, iar a doua impară și prima coordonată impară, iar a doua pară). Se vor număra punctele din fiecare categorie și se va calcula pe baza unei formule simple modul în care se pot alege puncte din diferite categorii pentru a obține un determinant par.

11. Pentru fiecare particulă radioactivă se va calcula tangenta unghiului format de dreapta care o unește de origine cu axa orizontală. Se vor alege apoi particulele pentru care se obține cea mai mică și cea mai mare valoare a tangentei și se va determina unghiul format de dreptele care unesc cele două particule de origine. Trebuie observat faptul că în locul tangentei se poate utiliza orice altă funcție f care păstrează ordinea (în cazul în care $a < b$, avem întotdeauna fie $f(a) < f(b)$, fie $f(b) < f(a)$). Așadar, se pot utiliza funcțiile sinus sau cosinus, dar și altele care satisfac proprietatea amintită.

12. Problema cere să se determine o dreaptă care împarte o mulțime de puncte în două mulțimi cu același cardinal. Pentru a rezolva problema, se sortează punctele în funcție de coordonata x și se ia ca soluție dreapta verticală care trece prin punctul cu indexul n din șirul sortat al punctelor. Este evident că această dreaptă conține cel puțin n puncte pe ea și în stânga ei, dar mai conține cel puțin n puncte pe ea și în dreapta ei.

13. Dacă un punct se află în interiorul piramidei, pe una din fețe, pe una din muchii sau este un vârf, atunci suma volumelor celor patru piramide determinate de punctul respectiv și grupuri de trei vârfuri ale piramidei este egală cu volumul piramidei. Volumul unei piramide se poate afla cu formula $(Arie_{baza} \cdot H)/3$.

14. Algoritmul de rezolvare al acestei probleme este foarte simplu: se pornește cu cel mai mare număr $(2 \cdot n)$ și se caută perechea lui, parcurgând valorile în ordine descrescătoare. După determinarea numărului pereche, acesta se marchează ca fiind utilizat. Se continuă cu următoarele valori neutilizate $(2 \cdot n - 1, 2 \cdot n - 2, \dots)$, pentru care se determină perechile în același mod. Numerele pereche determinate la fiecare pas se marchează ca fiind utilizate și nu mai sunt folosite pentru determinarea unei alte perechi. În final, se vor obține toate cele n perechi. Deși nu se cunoaște o demonstrație matematică a acestui fapt, algoritmul descris determină toate perechile cel puțin pentru toate valorile n cuprinse între 1 și 16383. Parcurgând valorile în ordine crescătoare (de la 1 la $2 \cdot n$), nu sunt determinate toate cele n perechi.

15. La fiecare pas se scoate un element din lista și trebuie să se găsească un element cu un index determinat (dacă elementul eliminat la pasul curent k era la poziția i , atunci elementul pentru eliminare la pasul următor este pe poziția $(i-1+k) \bmod (n-i+1)$). Pentru a realiza această operație eficient, se va folosi un șir $Nr[i]$ cu semnificația: câți elevi cu numărul de ordine între i și $i+\sqrt{n}$ mai sunt în listă. Pentru a elimina un element de index p din listă, trebuie actualizați indecșii $p-\sqrt{n} \dots p$ ai lui Nr . Pentru a găsi elementul al p -lea ce nu a fost încă eliminat, se parcurge în salturi de \sqrt{n} și se ajunge la un index i , astfel încât al p -lea element se află între i și $i+\sqrt{n}$, deci pentru găsirea celui de-al p -lea element ce nu a fost încă eliminat se fac maxim $2 \cdot \sqrt{n}$ operații.

16. Se construiește o matrice auxiliară C astfel încât elementul de pe poziția (i,j) va conține, în cazul în care în labirint la această poziție se poate ajunge și nu este zid, lungimea minimă a drumului de la ea până la poziția de plecare. Inițial, toate elementele vor fi inițializate cu -1, cu excepția punctului de plecare, care va fi inițializat cu 0. Până în momentul în care se modifică valoarea unui element aflat pe prima linie sau pe prima coloană sau pe ultima linie sau pe ultima coloană, fiecare element din matrice va primi valoarea minimă a vecinilor la care se adaugă 1. În momentul în care se modifică un element de pe margine, s-a găsit o soluție. Pentru a reconstitui drumul se va folosi o rutină recursivă, determinând la fiecare pas poziția vecinului unui punct (i,j) din matrice, pentru care valoarea este $C[i,j]-1$; apelurile recursive se încheie când se ajunge la punctul inițial.

17. Pentru a afla probabilitatea cerută, trebuie aflat câte numere există în intervalul $[A..B]$ cu cel puțin K cifre de C . Pentru asta se va construi o funcție $f(x)$ care va returna câte numere sunt în intervalul $[0..x-1]$ care au cel puțin K cifre de C . Astfel, rezultatul va fi $f(B+1)-f(A)$. Funcția $f(x)$ va parcurge numărul x cifră cu cifră, adunând la fiecare pas câte numere există care respectă condițiile impuse și au ca prefix cifrele din x fixate până acum. Ca să se realizeze acest lucru este nevoie de următoarele informații:

$Nr(i, j)$ = câte numere între 0 și $10^i - 1$ conțin j cifre de C (se va considera că numerele pot avea 0-uri în față, de exemplu: 0003 are 3 cifre de 0).

Acest număr se poate calcula cu o relație de recurență (independentă de variabila C): $Nr(i, j) = 9 * Nr(i-1, j) + Nr(i-1, j-1)$.

De asemenea, mai este nevoie de următoarele informații:

$Nr0(i, j)$ = câte numere care nu au 0-uri în față între 0 și $10^i - 1$ conțin j cifre de C .

Relația de recurență va fi în funcție de C :

dacă $C=0$ atunci $Nr0(i, j) \leftarrow Nr0(i-1, j) + 9 * Nr(i-1, j)$
 altfel $Nr0(i, j) \leftarrow Nr0(i-1, j) + 8 * Nr(i-1, j) + Nr(i-1, j-1)$

Odată disponibile aceste informații, se poate realiza destul de ușor funcția $f(x)$, acordând atenție la cazurile care pot apărea.

18. Cum triunghiurile echilaterale au toate unghiurile și toate laturile egale, se poate calcula ușor, folosind formule matematice, unde se află al treilea punct pentru a forma un triunghi echilateral, dacă se fixează două puncte. Astfel, rezolvarea se reduce la încercarea fiecărei pereche de puncte, la calcularea celui de-al treilea punct, și la verificarea existenței acestuia; pentru a realiza această verificare eficient, se sortează punctele la începutul programului, iar când se va căuta un punct se va realiza o căutare binară.

19. Se va folosi o procedură recursivă care determină în câți pași se ajunge la coordonatele (x, y) , dacă punctele sunt parcurse în ordinea dată de curba de ordin k . Se împarte curba de ordinul k în patru curbe de ordinul $k-1$ și se determină din ce curbă provine (x, y) , apoi se face un apel recursiv. Procedura se oprește când se ajunge la o curbă de ordinul 1.

20. Nu se pot genera toate primele P fracții datorită limitei mari pentru P , așadar se va căuta binar numărătorul celei de a P -a fracție, pe baza faptului că numărul de fracții ireductibile cu numitorul N și cu numărătorul $\leq x$ fixat, crește odată cu creșterea numărului x . Așadar, problema se reduce la determinarea numărului de fracții ireductibile cu numitorul N , cu numărătorul $\leq x$ fixat. Pentru a realiza acest lucru, se vor număra câte fracții redutibile cu numitorul N și cu numărătorul $\leq x$ există, folosind principiul includerii și excluderii:

$$\left| \bigcup_{i=1}^n A_i \right| = \sum_{i=1}^n |A_i| - \sum_{1 \leq i < j \leq n} |A_i \cap A_j| + \sum_{1 \leq i < j < k \leq n} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n-1} \cdot \left| \bigcap_{i=1}^n A_i \right|$$

Mulțimile A_i vor reprezenta mulțimile de numărători divizibili cu al i -lea număr prim care este divizor al lui N (se iau divizorii lui N pentru a obține fracții redutibile – este evident că nu sunt necesari toți divizorii, ci numai cei primi – sunt maxim 10 pentru limita lui N din enunț).

21. Se vor construi doi vectori:

$P[i]$ = numărul minim de persoane necesare pentru a obține eficiența i ;

$S[i]$ = numărul minim de șefi necesari pentru a obține eficiența i .

Se iau toate grupurile posibile de x persoane, dintre care y șefi și se actualizează valorile $P[i+(x-y)*y]$ și $S[i+(x-y)*y]$ în funcție de $P[i]$ și $S[i]$.

22. Problema cere determinarea numărului de permutări de lungime N care conțin K secvențe monotone crescătoare. Se va nota acest număr cu $A(n, k)$. Într-o permutare de lungime $N-1$ cu K secvențe monotone, elementul cu valoare N poate fi introdus în K locuri și anume la sfârșitul fiecărei secvențe monotone, pentru a forma o permutare de lungime N și să existe tot K secvențe monotone. Într-o permutare de lungime $N-1$ cu $K-1$ secvențe monotone, se poate insera ca înainte elementul N în $K-1$ locuri pentru a obține tot $K-1$ secvențe monotone, deci în celelalte $N-K+1$ locuri, inserarea lui N va produce o permutare de lungime N cu K secvențe monotone, deoarece se va sparge o secvență monotona în două secvențe monotone mai mici. Astfel, se obține următoarea relație de recurență:

$$A(n, k) = k * A(n-1, k) + (n-k+1) * A(n-1, k-1).$$

23. Numărul submulțimilor mulțimii $\{1, 2, \dots, n\}$ este 2^n dacă se contorizează și mulțimea vidă. Numărul submulțimilor mulțimii $\{1, 2, \dots, n\}$ care încep cu un număr oarecare x sunt în număr de 2^{n-x} . Numărul de ordine al unei submulțimi a mulțimii $\{1, 2, \dots, n\}$ care începe cu x este cel puțin $2^{n-1} + 2^{n-2} + \dots + 2^{n-x+1}$ și cel mult $2^{n-1} + 2^{n-2} + \dots + 2^{n-x} - 1$. Aceleași raționamente se aplică pentru submulțimea rămasă. Aceasta va trebui să fie o submulțime validă a mulțimii $\{1, 2, \dots, n-1\}$. Pentru a realiza acest lucru, se scade câte o unitate din fiecare număr al submulțimii rămase. Această operație este posibilă în cazul în care se consideră că elementele unei submulțimi sunt ordonate crescător. Folosind metoda descrisă mai sus, se poate determina numărul de ordine al unei submulțimi. Folosind un raționament asemănător, se poate realiza și operația inversă.

24. Pentru a închide prima fereastră deschisă, trebuie să se execute 1 click + numărul de click-uri necesare pentru a închide ferestrele deschise după prima fereastră și care acoperă colțul din dreapta-sus al ferestrei.

25. În $Sol[J, F, M, A]$ se va reține valoarea maximă a unei echipe formate din F fundași, M mijlocași și A atacanți, folosind numai primii J jucători. Pentru stabilirea valorilor $Sol[J]$ este nevoie de valorile $Sol[J-1]$. La sfârșit, soluția va fi în $Sol[N, 4, 4, 2]$.

26. Algoritmul de rezolvare al problemei este următorul:

- Se va lua fiecare punct în parte, considerându-l centrul axelor de coordonate. Fie M acest punct și (M_x, M_y) coordonatele sale inițiale. Pentru fiecare punct P , coordonatele sale vor deveni $(P_x - M_x, P_y - M_y)$;
- Se consideră, pentru fiecare punct P , unghiul pe care segmentul MP îl formează cu axa Ox ;
- Se sortează punctele după valoarea unghiului calculat la pasul precedent
- În șirul astfel obținut, cea mai mare secvență de puncte cu unghiul caracteristic identic, împreună cu punctul M va reprezenta mulțimea maximală de puncte coliniare din care face parte M .

27. Soluția constă în generarea acționărilor de comutatoare de pe prima linie a panoului. În acest mod, pe a doua linie vor trebui acționate doar acele comutatoare care vor stinge becurile de pe prima linie. Se va parcurge astfel panoul până se efectuează acționările de pe ultima linie. Dacă toate becurile de pe ultima linie sunt stinse, înseamnă că s-a găsit o soluție. Dintre toate aceste soluții, se va păstra aceea cu număr minim de acționări.

28. Numărul N are maxim $2 \cdot N^{1/2}$ divizori. Dacă se descompune numărul N în factori primi, atunci se poate scrie în forma $P_1^{E_1} \cdot P_2^{E_2} \cdot \dots \cdot P_k^{E_k}$, unde P_1, P_2, \dots, P_k sunt numere prime și E_1, E_2, \dots, E_k sunt numere naturale mai mari ca zero. Un divizor al lui N va fi reprezentat printr-un vector de exponenți (e_1, e_2, \dots, e_k) , unde $0 \leq e_i \leq E_i$. Prin urmare, problema poate fi ușor redusă la următoarea cerință: ordonați toți vectorii (e_1, e_2, \dots, e_k) , unde $0 \leq e_i \leq E_i$, într-un șir cu proprietatea că diferența între doi vectori consecutivi se realizează la o singură poziție a vectorilor și cele două elemente ale vectorilor de pe poziția respectivă diferă prin o unitate. Această problemă este o generalizare a determinării codului Gray și poate fi rezolvată modificând una din metodele folosite pentru a genera codul Gray. Se presupune că se știe să se genereze o soluție pentru $M = N / (P_k^{E_k})$ (o soluție pentru un număr cu $k-1$ factori primi). Fie C , vectorul soluție pentru M și R , vectorul C inversat (primul element al lui R este ultimul al lui C , etc.). O soluție pentru N poate fi obținută în următoarea formă: $C, P_k \cdot R, P_k^2 \cdot C, P_k^3 \cdot R, \dots$

Astfel, s-au concatenat E_k coduri pentru M , înmulțind cu P_k și pe fiecare poziție impară s-a pus secvența inversată pentru M .

29. Pentru a rezolva această problemă, la început, se vor genera primele 8 linii:

```

1
1 1
1 0 1
1 1 1 1
1 0 0 0 1
1 1 0 0 1 1
1 0 1 0 1 0 1
1 1 1 1 1 1 1 1

```

Completăm toate liniile cu zerouri până la cea mai apropiată putere a lui 2:

```

1
1 1
1 0 1 0
1 1 1 1
1 0 0 0 1 0 0 0
1 1 0 0 1 1 0 0
1 0 1 0 1 0 1 0
1 1 1 1 1 1 1 1

```

Se observă că jumătatea a doua a fiecărei linii este identică cu prima jumătate. Mai mult, elementele din prima jumătate a unui rând cu 2^k elemente (după completare) sunt identice cu elementele aflate cu 2^{k-1} rânduri mai sus (dacă s-ar completa și rândurile respective cu suficiente zerouri). Pentru a valida aceste observații, se poate demonstra prin inducție că observațiile sunt valabile în tot tabelul. În acest moment, ideea de rezolvare devine destul de clară: pentru a genera linia N se generează și se dublează "prima jumătate" (este vorba despre jumătatea după completarea până la o putere a lui 2).

Pentru a genera "prima jumătate" se apelează recursiv aceeași metodă, având ca parametru un nou N , care se obține scăzând din vechiul N cea mai mare putere a lui 2, mai mică decât el. Există și alte variante de rezolvare eficiente, toate exploatând proprietățile descrise mai sus.